University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Fachbereich Technik
Abteilung Elektrotechnik und Informatik

PROGRAMMIEREN III
AUFGABE 3

ENTWICKLUNG EINES WARENWIRTSCHAFTSSYSTEMS: PRODUKTE IN REGALEN PRÄSENTIERT, EINKAUFEN UND KASSIEREN IN C++

Gruppe A1
Studiengang Elektrotechnik
Vorgelegt von

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

Emden, 16. November 2023

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

Kapitel 1

Quellencode

1.1 Datei ’main.cc’

Listing 1.1 Die main.cc

```
/**
 * @file main.cc
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Hauptprogramm fuer das Lagerverwaltungssystem.
 * @version 0.3
 * @date 2023-11-13
 *
 * Diese Datei dient als Einstiegspunkt fuer das Lagerverwaltungssystem. Sie
 * liest Befehlszeilenargumente,
 * initialisiert das Lager und die Regale, ermoeglicht dem Benutzer das
 * Einkaufen und fuehrt die entsprechenden Operationen aus.
 *
 * @copyright Copyright (c) 2023
 *
 */
#include "kasse.hh"
#include "laden.hh"
#include "lager.hh"
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <map>
#include <ostream>
#include <string>
#include <vector>

using namespace std;

/**
 * @def CLEAR
 * @brief Definition fuer den Befehl zum Loeschen des Konsolenbildschirms.
 */
#define CLEAR u8"\033[2J\033[1;1H"

/**
 * @brief Hauptfunktion des Programms.
 *
 * Diese Funktion dient als Einstiegspunkt fuer das Lagerverwaltungssystem. Sie
 * liest Befehlszeilenargumente, initialisiert das Lager und die Regale,
 * ermoeglicht dem Benutzer das Einkaufen und fuehrt die entsprechenden
 * Operationen aus.
 *
 * @param argc Die Anzahl der Befehlszeilenargumente.
 * @param argv Ein Array von Zeichenketten, das die Befehlszeilenargumente
 * enthaelt.
 * @return Eine Ganzzahl, die den Programmstatus zurueckgibt (0 fuer Erfolg,
```

```
 * andere Werte fuer Fehler).
 */
int main(int argc, char *argv[]) {
  string filewrite = "";
  string fileread = "";
  char wahl;
  string vorname;
  string name;
  Lager lager;
  vector<Regal> regale;
  for (int i = 1; i < argc; i++) {
    string arg = argv[i];
    if (arg == "-o") {
      filewrite = argv[i + 1];
    } else if (string(argv[i]) == "-i") {
      fileread = argv[i + 1];
    }
  }

  if (fileread == "") {
    exit(EXIT_FAILURE);
  }

  // Lager aus der Eingabedatei lesen
  lager.readFile(fileread);

  // Regale initialisieren
  Regal gemueseRegal(R"(Gemuese)", lager, {40, 41});
  Regal getraenkeRegal("Getraenke", lager, {43, 50, 55});
  Regal sonderRegal("Sonderartikel", lager, 10);

  regale.push_back(gemueseRegal);
  regale.push_back(getraenkeRegal);
  regale.push_back(sonderRegal);

  // Hauptbenutzerschleife
  while (true) {
    cout << CLEAR;
    cout << "Waehlen Sie aus!" << endl;
    cout << setw(20) << left << "\tEinkaufen: "
         << "n" << endl;
    cout << setw(20) << left << "\tFeierabend: "
         << "q" << endl;
    cout << "\nAuswahl:";
    cin >> wahl;

    // Programm beenden, wenn 'q' ausgewaehlt wird
    if (wahl == 'q') {
      break;
    }

    // Einkaufen starten, wenn 'n' ausgewaehlt wird
    if (wahl == 'n') {
      cout << "Geben Ihre Name!" << endl;
      cin >> vorname;
      cin >> name;
      Kunde kunde(vorname + string(" ") + name, regale);
      kunde.kundeUI();
      cout << CLEAR;
    } else {
      break;
    }
  }

  // Lager aktualisieren und speichern
  if (filewrite == "") {
    lager.write("out.txt");
  } else {
    lager.write(filewrite);
  }
  return 0;
}
```

1.2 Datei ’Lager.hh’

Listing 1.2 Die Header-Datei lager.hh

```
/**
 * @file lager.hh
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Definitionen der Lagerverwaltungsfunktionen.
 * @version 0.3
 * @date 2023-11-13
 *
 * Dieses Header-Datei enthaelt die Definitionen von Klassen und Funktionen zur
 * Verwaltung von Artikeln und Warengruppen in einem C++-Programm.
 *
 * @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;
typedef int Nummer;

/**
 * @brief Die Klasse "Artikel" repraesentiert einen Artikel mit verschiedenen
 * Eigenschaften.
 */
class Artikel {
protected:
  string artikelname;
  string artikelnummer;
  unsigned int lagerbestand;
  masseinheit einheit;
  preis verkaufpreis;
  preis normpreis;

public:
  /**
   * @brief Standardkonstruktor fuer die Klasse "Artikel".
   */
  Artikel();

  /**
   * @brief Konstruktor fuer die Klasse "Artikel".
   *
   * @param name Der Name des Artikels.
   * @param num Die Artikelnummer des Artikels.
   * @param bestand Der Lagerbestand des Artikels.
   * @param einheit Die Einheit des Artikels (stk, kg, l).
   * @param vp Der Verkaufspreis des Artikels.
   * @param np Der Normalpreis des Artikels.
   */
  Artikel(string name, string num, unsigned int bestand, masseinheit einheit,
          preis vp, preis np);

  // Getter-Funktionen

  /**
   * @brief Destruktor fuer die Klasse "Artikel".
   */
  ~Artikel();

  /**
   * @brief Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam
   * genutzt wird.
   */
```

```
70  // static Warengruppen gruppe;
71
72  /**
73   * @brief Setzt die Warengruppe fuer Artikel.
74   *
75   * @param g Die Warengruppe, die zugewiesen werden soll.
76   */
77  // static void setGruppe(Warengruppen g);
78
79  /**
80   * @brief Gibt den Namen des Artikels zurueck.
81   *
82   * @return Der Name des Artikels.
83   */
84  string getName() const;
85
86  /**
87   * @brief Gibt die Artikelnummer des Artikels zurueck.
88   *
89   * @return Die Artikelnummer des Artikels.
90   */
91  string getArtikelnummer() const;
92
93  /**
94   * @brief Gibt den Lagerbestand des Artikels zurueck.
95   *
96   * @return Der Lagerbestand des Artikels.
97   */
98  unsigned int getLagerbestand() const;
99
100 /**
101  * @brief Gibt die Masseinheit des Artikels als Wert aus der Enumeration
102  * zurueck.
103  *
104  * @return Die Masseinheit des Artikels als Wert aus der Enumeration (stk, kg,
105  * l).
106  */
107 masseinheit getMasseinheit() const;
108
109 /**
110  * @brief Gibt die Masseinheit des Artikels als Zeichenkette zurueck.
111  *
112  * @return Die Masseinheit des Artikels als Zeichenkette ("stk", "kg", "l").
113  */
114 string getStrMasseinheit() const;
115
116 /**
117  * @brief Gibt den Verkaufspreis des Artikels zurueck.
118  *
119  * @return Der Verkaufspreis des Artikels.
120  */
121 preis getVerkaufpreis() const;
122
123 /**
124  * @brief Gibt den Normalpreis des Artikels zurueck.
125  *
126  * @return Der Normalpreis des Artikels.
127  */
128 preis getNormpreis() const;
129
130 /**
131  * @brief Gibt die Warengruppe des Artikels zurueck.
132  *
133  * @return Die Warengruppe des Artikels oder die Artikelnummer, falls keine
134  * Warengruppe gefunden wurde.
135  */
136 int getGruppe() const;
137
138 // Setter-Funktionen
139
140 /**
141  * @brief Setzt den Namen des Artikels.
142  *
```

```
143     * @param name Der neue Name des Artikels.
144     */
145    void setName(string name);
146
147    /**
148     * @brief Setzt die Artikelnummer des Artikels.
149     *
150     * @param num Die neue Artikelnummer des Artikels.
151     */
152    void setArtikelnummer(string num);
153
154    /**
155     * @brief Setzt den Lagerbestand des Artikels.
156     *
157     * @param bestand Der neue Lagerbestand des Artikels.
158     */
159    void setLagerbestand(unsigned int bestand);
160
161    /**
162     * @brief Setzt die Masseinheit des Artikels.
163     *
164     * @param einheit Die neue Masseinheit des Artikels (stk, kg, l).
165     */
166    void setMasseinheit(masseinheit einheit);
167
168    /**
169     * @brief Setzt den Verkaufspreis des Artikels.
170     *
171     * @param vp Der neue Verkaufspreis des Artikels.
172     */
173    void setVerkaufpreis(preis vp);
174
175    /**
176     * @brief Setzt den Normalpreis des Artikels.
177     *
178     * @param np Der neue Normalpreis des Artikels.
179     */
180    void setNormpreis(preis np);
181
182    /**
183     * @brief Gibt die Artikelinformationen aus.
184     *
185     * Diese Funktion gibt die Informationen des Artikels aus, einschliesslich
186     * Artikelname, Artikelnummer, Lagerbestand, Verkaufspreis, Masseinheit und
187     * Normpreis.
188     *
189     * @param os Die Ausgabestromreferenz, in die die Informationen geschrieben
190     * werden.
191     * @return Die Ausgabestromreferenz, in die die Informationen geschrieben
192     * wurden.
193     */
194    ostream &print(ostream &outstream);
195  };
196  /**
197   * @brief ueberladen des Ausgabeoperators fuer die Artikelklasse.
198   *
199   * Diese Funktion ermoeglicht das Ausgeben eines Artikels mit dem
200   * Ausgabeoperator
201   * '<<'.
202   *
203   * @param os Die Ausgabestromreferenz, in die die Informationen geschrieben
204   * werden.
205   * @param produkt Der Artikel, der ausgegeben werden soll.
206   * @return Die Ausgabestromreferenz, in die die Informationen geschrieben
207   * wurden.
208   */
209  ostream &operator<<(ostream &os, Artikel produkt);
210
211  /**
212   * @brief ueberladen des Eingabeoperators fuer die Artikelklasse.
213   *
214   * Diese Funktion ermoeglicht das Einlesen von Artikelinformationen mit dem
215   * Eingabeoperator '>>'.
```

```
 *
 * @param is Die Eingabestromreferenz, aus der die Informationen eingelesen
 * werden.
 * @param produkt Der Artikel, in den die Informationen eingelesen werden
 * sollen.
 */
void operator>>(istream &is, Artikel &produkt);

/**
 * @brief Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert
 * sie fuer Stueckgut-Artikel.
 */
class Stueckgut : public Artikel {
private:
public:
  /**
   * @brief Konstruktor fuer die Klasse "Stueckgut".
   *
   * @param name Der Name des Stueckgut-Artikels.
   * @param num Die Artikelnummer des Stueckgut-Artikels.
   * @param vp Der Verkaufspreis des Stueckgut-Artikels.
   * @param bestand Der Lagerbestand des Stueckgut-Artikels (Standardwert: 1).
   */
  Stueckgut(Artikel produkt);
  Stueckgut(string name, string num, preis vp, unsigned int bestand = 1);
};

/**
 * Die Klasse "Schuettgut" erbt von der Klasse "Artikel" und spezialisiert sie
 * fuer Schuettgut-Artikel.
 * Die Klasse "Schuettgut" erweitert die Basisfunktionalitaet der Klasse
 * "Artikel" um die Beruecksichtigung der Losgroesse.
 */
class Schuettgut : public Artikel {
private:
  double losgroesse;

public:
  /**
   * @brief Konstruktor fuer die Klasse "Schuettgut" unter Verwendung eines
   * bereits existierenden Artikels.
   *
   * @param produkt Der Artikel, aus dem ein Schuettgut-Artikel erstellt wird.
   */
  Schuettgut(Artikel produkt);

  /**
   * @brief Konstruktor fuer die Klasse "Schuettgut".
   *
   * @param name Der Name des Schuettgut-Artikels.
   * @param num Die Artikelnummer des Schuettgut-Artikels.
   * @param groesse Die Losgroesse des Schuettgut-Artikels.
   * @param np Der Normalpreis des Schuettgut-Artikels.
   * @param bestand Der Lagerbestand des Schuettgut-Artikels (Standardwert: 1).
   */
  Schuettgut(string name, string num, double groesse, preis np,
             unsigned int bestand = 1);

  /**
   * @brief Gibt die Losgroesse des Schuettgut-Artikels zurueck.
   *
   * @return Die Losgroesse des Artikels.
   */
  double getLosgroesse() const;

  /**
   * @brief Setzt den Verkaufspreis des Schuettgut-Artikels basierend auf der
   * Losgroesse.
   *
   * @param vp Der Verkaufspreis, der gesetzt werden soll.
   */
  void setVerkaufpreis(preis vp);

```

```
289   /**
290    * @brief Setzt die Losgroesse des Schuettgut-Artikels.
291    *
292    * @param groesse Die neue Losgroesse.
293    */
294   void setLosgroesse(double groesse);
295 };
296
297 /**
298  * @brief Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und
299  * spezialisiert sie fuer Fluessigkeits-Artikel.
300  *
301  * Die Klasse "Fluessigkeit" erweitert die Basisfunktionalitaet der Klasse
302  * "Artikel" um die Beruecksichtigung des Volumens.
303  */
304 class Fluessigkeit : public Artikel {
305 private:
306   double volume;
307
308 public:
309   /**
310    * @brief Konstruktor fuer die Klasse "Fluessigkeit" unter Verwendung eines
311    * bereits existierenden Artikels.
312    *
313    * @param produkt Der Artikel, aus dem ein Fluessigkeits-Artikel erstellt
314    * wird.
315    */
316   Fluessigkeit(Artikel produkt);
317
318   /**
319    * @brief Konstruktor fuer die Klasse "Fluessigkeit".
320    *
321    * @param name Der Name des Fluessigkeits-Artikels.
322    * @param num Die Artikelnummer des Fluessigkeits-Artikels.
323    * @param vol Das Volumen des Fluessigkeits-Artikels.
324    * @param np Der Normalpreis des Fluessigkeits-Artikels.
325    * @param bestand Der Lagerbestand des Fluessigkeits-Artikels (Standardwert:
326    * 1).
327    */
328   Fluessigkeit(string name, string num, double vol, preis np,
329                unsigned int bestand = 1);
330
331   /**
332    * @brief Gibt das Volumen des Fluessigkeits-Artikels zurueck.
333    *
334    * @return Das Volumen des Artikels.
335    */
336   double getVolume() const;
337
338   /**
339    *
340    * @param vp Der Verkaufspreis, der gesetzt werden soll.
341    */
342   void setVerkaufpreis(preis vp);
343
344   /**
345    * @brief Setzt das Volumen des Fluessigkeits-Artikels.
346    *
347    * @param vol Das neue Volumen.
348    */
349   void setVolume(double vol);
350 };
351 /**
352  * @class Lager
353  * @brief Klasse, die ein Lagerverwaltungssystem repraesentiert.
354  */
355 class Lager {
356 public:
357   /**
358    * @brief Standardkonstruktor fuer die Klasse Lager.
359    */
360   Lager() = default;
361
```

```
362   /**
363    * @brief Destruktor fuer die Klasse Lager.
364    *
365    * Der Destruktor durchlaeuft die Lager-Map und gibt den zugewiesenen Speicher
366    * fuer jeden Artikel frei, bevor das Lager-Objekt zerstoert wird.
367    */
368   ~Lager();
369
370   /**
371    * @brief Typdefinition fuer eine Map von Artikelnummern zu Artikeln.
372    */
373   typedef map<string, Artikel *> artikelMap;
374
375   /**
376    * @brief Liest Artikelinformationen aus einer Datei und fuegt sie dem Lager
377    * hinzu.
378    * @param filename Der Dateiname der Eingabedatei.
379    */
380   void readFile(string filename);
381
382   /**
383    * @brief Schreibt die Artikelinformationen in den angegebenen Ausgabeostream.
384    * @param os Der Ausgabeostream, in den die Informationen geschrieben werden.
385    */
386   void write(ostream &os);
387
388   /**
389    * @brief Schreibt die Artikelinformationen in eine Datei.
390    * @param filename Der Dateiname der Ausgabedatei.
391    */
392   void write(string filename);
393
394   /**
395    * @brief Gibt den Artikel mit der angegebenen Artikelnummer zurueck.
396    * @param artikelnummer Die Artikelnummer des gesuchten Artikels.
397    * @return Der Artikel mit der angegebenen Artikelnummer.
398    */
399   Artikel getArtikel(string artikelnummer) const;
400
401   /**
402    * @brief Gibt die gesamte Map von Artikelnummern zu Artikeln zurueck.
403    * @return Die Map von Artikelnummern zu Artikeln.
404    */
405   artikelMap getMap();
406
407   /**
408    * @brief Aktualisiert die Informationen fuer einen Artikel in der Map.
409    * @param num Die Artikelnummer des zu aktualisierenden Artikels.
410    * @param artikel Der aktualisierte Artikel.
411    */
412   void updateArtikel(string num, Artikel *artikel);
413
414 private:
415   ///< Die Map von Artikelnummern zu Artikeln im Lager.
416   artikelMap lagerMap;
417 };
418 #endif // !LAGER_HH
```

1.3 Datei 'Lager.cc'

Listing 1.3 Die Header-Datei lager.cc

```
/**
 * @file lager.cc
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Implementierung der Lagerverwaltungsfunktionen.
 * @version 0.3
 * @date 2023-11-13
 *
 * 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 <fstream>
#include <iostream>
#include <map>
#include <sstream>
#include <string>
#include <vector>

static double rounding(double);

Lager::~Lager() {
  artikelMap::iterator it = lagerMap.begin();
  while (it != lagerMap.end()) {
    delete (it->second);
    it++;
  }
}

void Lager::readFile(string filename) {
  ifstream file(filename);
  if (file.is_open()) {
    Artikel tmp;
    do {
      try {
        file >> tmp;
        switch (tmp.getMasseinheit()) {
        case 0:
          lagerMap.insert({tmp.getArtikelnummer(), new Stueckgut(tmp)});
          break;
        case 1:
          lagerMap.insert({tmp.getArtikelnummer(), new Schuettgut(tmp)});
          break;
        case 2:
          lagerMap.insert({tmp.getArtikelnummer(), new Fluessigkeit(tmp)});
          break;
        }
      } catch (const int &ex) {
      } catch (std::invalid_argument const &ex) {
      }
    } while (!file.eof());
    file.close();
  } else {
    exit(EXIT_FAILURE);
  }
}

void Lager::write(ostream &os) {
  artikelMap::iterator it = lagerMap.begin();
  while (it != lagerMap.end()) {
    os << *it->second << endl;
    it++;
  }
}

```

```
70  void Lager::updateArtikel(string num, Artikel *artikel) {
71    delete (lagerMap[num]);
72    lagerMap[num] = artikel;
73  }
74
75  void Lager::write(string filename) {
76    ofstream file(filename);
77    if (file.is_open()) {
78      artikelMap::iterator it = lagerMap.begin();
79      while (it != lagerMap.end()) {
80        file << *it->second << endl;
81        it++;
82      }
83    }
84  }
85
86  Artikel Lager::getArtikel(string artikelnummer) const {
87
88    return *artikelMap(lagerMap)[artikelnummer];
89  }
90
91  Lager::artikelMap Lager::getMap() { return lagerMap; }
92
93  Artikel::Artikel() {}
94  Artikel::Artikel(string name, string num, unsigned int bestand,
95                   masseinheit einheit, preis vp, preis np)
96      : artikelname(name), artikelnummer(num), lagerbestand(bestand),
97        einheit(einheit), verkaufpreis(vp), normpreis(np) {}
98  Artikel::~Artikel() {}
99
100 // void Artikel::setGruppe(Warengruppen g) { gruppe = g; }
101 string Artikel::getName() const { return artikelname; }
102 string Artikel::getArtikelnummer() const { return artikelnummer; }
103 unsigned int Artikel::getLagerbestand() const { return lagerbestand; }
104 masseinheit Artikel::getMasseinheit() const { return einheit; }
105 string Artikel::getStrMasseinheit() const {
106   switch (einheit) {
107   case 0:
108     return "Stk";
109   case 1:
110     return "kg";
111   case 2:
112     return "l";
113   default:
114     return "None";
115   }
116 }
117
118 preis Artikel::getVerkaufpreis() const { return verkaufpreis; }
119 preis Artikel::getNormpreis() const { return normpreis; }
120 int Artikel::getGruppe() const {
121   string artnum = artikelnummer;
122   artnum = artnum.erase(4);
123   return stoi(artnum);
124 }
125
126 void Artikel::setName(string name) { artikelname = name; }
127 void Artikel::setArtikelnummer(string num) { artikelnummer = num; }
128 void Artikel::setLagerbestand(unsigned int bestand) { lagerbestand = bestand; }
129 void Artikel::setMasseinheit(masseinheit einheit) { this->einheit = einheit; }
130 void Artikel::setVerkaufpreis(preis vp) { verkaufpreis = vp; }
131 void Artikel::setNormpreis(preis np) { normpreis = np; }
132
133 std::ostream &Artikel::print(std::ostream &os) {
134   return os << artikelname << "|" << artikelnummer << "|" << lagerbestand << "|"
135             << verkaufpreis << "|" << getStrMasseinheit() << "|" << normpreis;
136 }
137
138 std::ostream &operator<<(std::ostream &os, Artikel produkt) {
139   return produkt.print(os);
140 }
141
142 void operator>>(istream &is, Artikel &produkt) {
```

```
143   vector<string> beschreibung;
144   string text, name, num;
145   int bestand = 0;
146   preis vp = 0, np = 0;
147   masseinheit einheit;
148
149   getline(is, text);
150   stringstream ss(text);
151
152   if (!text[0]) {
153     throw(-1);
154   }
155   text = "";
156   for (size_t i = 0; getline(ss, text, '|') && i < 6; i++) {
157     beschreibung.push_back(text);
158   }
159   if (beschreibung.size() < 5)
160     throw -1;
161   name = beschreibung[0];
162   num = beschreibung[1];
163
164   if (beschreibung[4] == "kg")
165     einheit = kg;
166   else if (beschreibung[4] == "l")
167     einheit = l;
168   else if (beschreibung[4] == "stk")
169     einheit = stk;
170   else {
171     einheit = stk;
172   }
173
174   for (size_t i = 1; i < 6; i++) {
175     for (size_t j = 0; j < beschreibung[i].length(); j++) {
176       if (beschreibung[i][j] == ' ') {
177         beschreibung[i].erase(beschreibung[i].begin() + j);
178         j--;
179       }
180     }
181   }
182
183   if (name == "" || num == "" || num.length() != 10) {
184     throw(-1);
185   }
186   if (beschreibung[3] == "" && beschreibung[4] == "") {
187     throw(-1);
188   }
189
190   if (beschreibung[2] != "") {
191     bestand = stoi(beschreibung[2]);
192   } else {
193     bestand = 0;
194   }
195   if (beschreibung[3] != "") {
196     vp = stof(beschreibung[3]);
197   }
198
199   if (beschreibung.size() > 5 && beschreibung[5] != "") {
200     np = stof(beschreibung[5]);
201   }
202   if (vp == 0)
203     vp = np;
204   if (np == 0)
205     np = vp;
206
207   produkt.setMasseinheit(einheit);
208   produkt.setName(beschreibung[0]);
209   produkt.setArtikelnummer(beschreibung[1]);
210   produkt.setLagerbestand(bestand);
211   produkt.setVerkaufpreis(vp);
212   produkt.setNormpreis(np);
213 }
214
215 Stueckgut::Stueckgut(Artikel produkt)
```

```
    : Stueckgut(produkt.getName(), produkt.getArtikelnummer(),
                produkt.getVerkaufpreis(), produkt.getLagerbestand()) {}
Stueckgut::Stueckgut(string name, string num, preis vp, unsigned int bestand)
    : Artikel(name, num, bestand, stk, vp, vp) {}

Schuettgut::Schuettgut(Artikel produkt)
    : Schuettgut(produkt.getName(), produkt.getArtikelnummer(),
                 rounding(produkt.getVerkaufpreis() / produkt.getNormpreis()),
                 produkt.getNormpreis(), produkt.getLagerbestand()) {}
Schuettgut::Schuettgut(string name, string num, double groesse, preis np,
                       unsigned int bestand)
    : Artikel(name, num, bestand, kg, (groesse * np), np), losgroesse(groesse) {
}
double Schuettgut::getLosgroesse() const { return losgroesse; }
void Schuettgut::setLosgroesse(double groesse) {
  losgroesse = groesse;
  verkaufpreis = rounding(verkaufpreis);
}
void Schuettgut::setVerkaufpreis(preis vp) {
  verkaufpreis = vp;
  losgroesse = rounding(losgroesse);
}

Fluessigkeit::Fluessigkeit(Artikel produkt)
    : Fluessigkeit(produkt.getName(), produkt.getArtikelnummer(),
                   rounding(produkt.getVerkaufpreis() / produkt.getNormpreis()),
                   produkt.getNormpreis(), produkt.getLagerbestand()) {}
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 = rounding(verkaufpreis);
}
void Fluessigkeit::setVerkaufpreis(preis vp) {
  verkaufpreis = vp;
  volume = rounding(volume);
}

static double rounding(double num) {
  num *= 100;
  num += 0.5;
  num = int(num);
  num /= 100;
  return num;
}
```

1.4 Datei ’Laden.hh’

Listing 1.4 Die Header-Datei lager.cc

```
/**
 * @file laden.hh
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Enthaelt die Deklaration der Klasse Regal und der Klasse Kunde.
 * @version 0.3
 * @date 2023-11-13
 *
 * Dieses Header-Datei enthaelt die Definitionen von Klassen und Funktionen zur
 * Verwaltung von Regale und Kunden und Warengruppen in einem C++-Programm.
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef LADEN_HH
#define LADEN_HH

#include "lager.hh"
#include <set>
#include <vector>

/**
 * @class Regal
 * @brief Repraesentiert ein Regal im Lager.
 *
 * Die Klasse Regal stellt Informationen ueber ein Regal im Lager zur
 * Verfuegung, einschliesslich des Regalnamens, des zugeordneten Lagers, der
 * zugehoerigen Warengruppen und der im Regal befindlichen Artikel.
 */
class Regal {
public:
  /**
   * @brief Konstruktor fuer ein Regal mit einer einzelnen Warengruppe.
   *
   * @param name Der Name des Regals.
   * @param lager Referenz auf das Lager, zu dem das Regal gehoert.
   * @param warengruppe Die Warengruppe, die dem Regal zugeordnet ist.
   */
  Regal(string name, Lager &lager, int warengruppe);

  /**
   * @brief Konstruktor fuer ein Regal mit mehreren Warengruppen.
   *
   * @param name Der Name des Regals.
   * @param lager Referenz auf das Lager, zu dem das Regal gehoert.
   * @param warengruppen Die Menge der Warengruppen, die dem Regal zugeordnet
   * sind.
   */
  Regal(string name, Lager &lager, std::set<int> warengruppen);

  /**
   * @brief Gibt eine Referenz auf das Lager zurueck, zu dem das Regal gehoert.
   *
   * @return Eine Referenz auf das Lager.
   */
  Lager &getLager();

  /**
   * @brief Gibt den Namen des Regals zurueck.
   *
   * @return Der Name des Regals.
   */
  string getName() const;

  /**
   * @brief Gibt die Menge der Warengruppen zurueck, die dem Regal zugeordnet
   * sind.
   *
   * @return Eine Menge von Warengruppen.
```

```
70      */
71    set<int> getWaren() const;
72
73    /**
74     * @brief Gibt einen Artikel im Regal anhand der Artikelnummer zurueck.
75     *
76     * @param num Die Artikelnummer des gesuchten Artikels.
77     * @return Der gefundene Artikel im Regal.
78     */
79    Artikel getArtikel(string num) const;
80
81    /**
82     * @brief Gibt eine Liste der Artikel im Regal zurueck.
83     *
84     * @return Ein Vektor von Artikelnummern im Regal.
85     */
86    vector<string> getImRegal() const;
87
88    /**
89     * @brief ueberschriebener Ausgabeoperator fuer die Klasse Regal.
90     *
91     * @param os Der Ausgabestrom.
92     * @param regal Das Regal, das ausgegeben werden soll.
93     * @return Der Ausgabestrom.
94     */
95    friend ostream &operator<<(ostream &os, Regal regal);
96
97  private:
98    ///< Der Name des Regals.
99    string regalname;
100   ///< Referenz auf das Lager, zu dem das Regal gehoert.
101   Lager &lager;
102   ///< Die Menge der Warengruppen, die dem Regal zugeordnet sind.
103   std::set<int> waren;
104 };
105
106 /**
107  * @class Kunde
108  * @brief Repraesentiert einen Kunden mit Einkaufsfunktionen.
109  *
110  * Die Klasse Kunde stellt einen Kunden dar, der Einkaufsaktionen in einem Lager
111  * durchfuehren kann. Ein Kunde hat einen Namen, eine Liste von Regalen, die er
112  * durchsuchen kann, und einen Warenkorb, um Artikel hinzuzufuegen.
113  */
114 class Kunde {
115 public:
116   /**
117    * @brief Konstruktor fuer einen Kunden mit einem Namen und einer Liste von
118    * Regalen.
119    *
120    * @param name Der Name des Kunden.
121    * @param regale Eine Referenz auf eine Liste von Regalen, die der Kunde
122    * durchsuchen kann.
123    */
124   Kunde(string name, vector<Regal> const &regale);
125
126   /**
127    * @brief Oeffnet die Benutzeroberflaeche des Kunden fuer Einkaufsaktionen.
128    */
129   void kundeUI();
130
131   /**
132    * @brief Gibt den Namen des Kunden zurueck.
133    *
134    * @return Der Name des Kunden.
135    */
136   string getName() const;
137
138   /**
139    * @brief Gibt die Artikel in einem bestimmten Regal aus.
140    *
141    * @param num Die Nummer des Regals, das durchsucht werden soll.
142    */
```

```
143   void printArtikel(int num);
144
145   /**
146    * @brief Gibt den aktuellen Warenkorb des Kunden aus.
147    */
148   void printWarenkorb();
149
150   /**
151    * @brief Struktur zur Darstellung von Waren im Warenkorb.
152    */
153   typedef struct {
154     string artikelnummer; ///< Die Artikelnummer.
155     double menge;         ///< Die Menge des Artikels im Warenkorb.
156   } waren;
157
158   /**
159    * @brief Gibt den aktuellen Warenkorb des Kunden zurueck.
160    *
161    * @return Ein Vektor von Waren im Warenkorb.
162    */
163   vector<waren> getWarenkorb() const;
164
165 private:
166   string name;
167   vector<Regal> const &regale;
168   vector<waren> warenkorb;
169 };
170
171 #endif // !LADEN_HH
```

1.5 Datei 'Laden.cc'

Listing 1.5 Die Header-Datei lager.cc

```
/**
 * @file laden.hh
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Enthaelt die Deklaration der Klasse Regal und der Klasse Kunde.
 * @version 0.3
 * @date 2023-11-13
 *
 * Dieses Header-Datei enthaelt die Definitionen von Klassen und Funktionen zur
 * Verwaltung von Regale und Kunden und Warengruppen in einem C++-Programm.
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "laden.hh"
#include "kasse.hh"
#include "lager.hh"
#include <iomanip>
#include <string>
#include <unistd.h>
#include <vector>

using namespace std;
#define CLEAR u8"\033[2J\033[1;1H"

Regal::Regal(string name, Lager &lager, int warengruppe)
    : regalname(name), lager(lager) {
  waren.insert(warengruppe);
}
Regal::Regal(string name, Lager &lager, std::set<int> warengruppen)
    : regalname(name), lager(lager) {
  waren.merge(warengruppen);
}

string Regal::getName() const { return regalname; }
std::set<int> Regal::getWaren() const { return waren; }

Artikel Regal::getArtikel(string num) const { return lager.getArtikel(num); }

vector<string> Regal::getImRegal() const {
  Lager::artikelMap map = lager.getMap();
  vector<string> imRegal;
  for (int ware : waren) {
    Lager::artikelMap::iterator it = map.begin();
    while (it != map.end()) {
      int num = (*it->second).getGruppe();
      num /= 100;
      if (ware == num) {
        imRegal.push_back(it->first);
        // cout << typeid(*it->second).name() << endl;
      }
      it++;
    }
  }
  return imRegal;
}

Lager &Regal::getLager() { return lager; }

ostream &operator<<(ostream &os, Regal regal) {
  vector<string> imRegal = regal.getImRegal();
  int i = 0;
  for (auto num : imRegal) {
    // cout << Lager(regal.lager).getArtikel(num) << endl;
    Artikel artikel = regal.lager.getArtikel(num);
    i++;
    cout << setw(5) << "";
    cout << i << setw(9) << ":" << left;
    cout << setw(30) << artikel.getName();
```

```
70      cout << setw(20) << artikel.getLagerbestand();
71      cout << artikel.getVerkaufpreis() << "/"
72           << artikel.getVerkaufpreis() / artikel.getNormpreis() << setw(20)
73           << artikel.getStrMasseinheit() << endl;
74    }
75    return os;
76  }
77
78  Kunde::Kunde(string name, vector<Regal> const &regale)
79      : name(name), regale(regale) {}
80
81  vector<Kunde::waren> Kunde::getWarenkorb() const { return warenkorb; }
82
83  string Kunde::getName() const { return name; }
84
85  void Kunde::kundeUI() {
86    string wahl;
87    size_t wahlNum;
88    cout << CLEAR;
89    int i = 0;
90    cout << "Warenkorb: " << warenkorb.size() << endl;
91    cout << "Waehlen Sie einen Regal aus\n" << left << endl;
92    cout << setw(2) << "";
93    cout << "Wahl" << setw(9) << ":" << left;
94    cout << setw(30) << "Bezeichnung" << endl;
95    cout << setw(5) << "";
96    cout << "0" << setw(9) << ":" << left;
97
98    cout << "Warenkorb" << left;
99    cout << endl;
100   for (auto regal : regale) {
101     i++;
102     cout << setw(5) << "";
103     cout << i << setw(9) << ":" << left;
104     cout << regal.getName() << endl;
105   }
106   cout << setw(5) << "";
107   cout << "q" << setw(9) << ":" << left;
108
109   cout << "Beenden" << left;
110   cout << endl;
111   cout << endl;
112   while (true) {
113     cout << "Auswahl: ";
114     cin >> wahl;
115     if (wahl[0] == 'q')
116       return;
117     try {
118       wahlNum = stoi(wahl);
119       if (wahlNum > regale.size()) {
120         cout << "Falsche Eingabe!" << endl;
121       } else if (wahlNum == 0) {
122         printWarenkorb();
123         break;
124       } else {
125         printArtikel(wahlNum - 1);
126         break;
127       }
128     } catch (const std::exception &) {
129       cout << "Falsche Eingabe!!" << endl;
130     }
131   }
132 }
133
134 void Kunde::printArtikel(int num) {
135   cout << CLEAR;
136   string wahl1, wahl2;
137   size_t wahl1Num;
138   double wahl2num;
139   cout << "Warenkorb: " << warenkorb.size() << endl;
140   cout << "Waehlen Sie einen Artikel aus\n" << left << endl;
141   cout << setw(15) << "";
142   cout << setw(30) << "Bezeichnung";
```

```
143   cout << setw(20) << "Lagerbestand";
144   cout << setw(20) << "Preis/Einheit" << endl;
145
146   cout << regale[num];
147   cout << setw(5) << "";
148   cout << "." << setw(9) << ":" << left;
149   cout << "Zurueck" << left;
150   cout << endl;
151   cout << setw(5) << "";
152   cout << "q" << setw(9) << ":" << left;
153   cout << "Beenden" << left;
154   cout << endl;
155   cout << endl;
156   while (true) {
157     cout << "Auswahl: ";
158     cin >> wahl1;
159     if (wahl1[0] == 'q') {
160       break;
161     }
162     if (wahl1[0] == '.') {
163       kundeUI();
164       break;
165     }
166     try {
167
168       wahl1Num = stoi(wahl1);
169       if (wahl1Num > Regal(regale[num]).getImRegal().size()) {
170       } else {
171         wahl1Num--;
172         string artnum = Regal(regale[num]).getImRegal()[wahl1Num];
173         Artikel artikel = Regal(regale[num]).getArtikel(artnum);
174         cout << artikel.getName() << endl;
175         cout << "Geben Sie die Menge" << endl;
176         cin >> wahl2;
177         wahl2num = stof(wahl2);
178         if (wahl2num <= artikel.getLagerbestand()) {
179           warenkorb.push_back({artnum, wahl2num});
180           cout << CLEAR;
181           float menge = warenkorb[warenkorb.size() - 1].menge;
182           cout << menge << " * " << artikel.getName() << endl;
183           sleep(1);
184           printArtikel(num);
185           break;
186         }
187       }
188     } catch (const std::exception &) {
189     }
190     cout << "Falsche Eingabe!" << endl;
191   }
192 }
193 void Kunde::printWarenkorb() {
194   string wahl;
195   int i = 0;
196   cout << CLEAR;
197   cout << "Warenkorb: " << warenkorb.size() << endl;
198   cout << "Waehlen Sie aus\n" << left << endl;
199   for (auto ware : warenkorb) {
200     Artikel artikel = regale[0].getArtikel(ware.artikelnummer);
201     i++;
202     cout.imbue(locale("de_DE.UTF-8"));
203     cout << setw(5) << "";
204     cout << i << setw(9) << ":" << left;
205     cout << setw(30) << artikel.getName();
206     cout << setw(6) << artikel.getVerkaufpreis() << "/" << setw(4)
207          << artikel.getVerkaufpreis() / artikel.getNormpreis();
208     cout << setw(20) << artikel.getStrMasseinheit();
209     cout << setw(20) << ware.menge;
210     cout << setw(20) << showbase << (artikel.getNormpreis() * ware.menge)
211          << endl;
212   }
213   cout << setw(5) << "";
214   cout << "k" << setw(9) << ":" << left;
215   cout << "Kasse" << left;
```

```
216   cout << endl;
217   cout << setw(5) << "";
218   cout << "." << setw(9) << ":" << left;
219   cout << "Zurueck" << left;
220   cout << endl;
221   cout << setw(5) << "";
222   cout << "q" << setw(9) << ":" << left;
223   cout << "Beenden" << left << endl;
224   cout << "Auswahl: ";
225   while (true) {
226     cin >> wahl;
227     if (wahl[0] == 'q') {
228       break;
229     }
230     if (wahl[0] == '.') {
231       kundeUI();
232       break;
233     }
234     if (wahl[0] == 'k') {
235       Kasse kasse(*this, Regal(regale[0]).getLager());
236       kasse.rechnung(cout);
237       break;
238     }
239     cout << "Falsche Eingabe!" << endl;
240   }
241 }
```

1.6 Datei 'Kasse.hh'

Listing 1.6 Die Header-Datei lager.cc

```
/**
 * @file laden.hh
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Enthaelt die Deklaration der Klasse Kasse.
 * @version 0.3
 * @date 2023-11-13
 *
 * Dieses Header-Datei enthaelt die Definitionen von Klassen und
 * Funktionen zur Verwaltung von der Kasse in einem C++-Programm.
 *
 * @copyright Copyright (c) 2023
 *
 */

#ifndef KASSE_HH
#define KASSE_HH

#include "laden.hh"
#include "lager.hh"
#include <iomanip>
#include <string>
#include <unistd.h>
#include <vector>

/**
 * @class Kasse
 * @brief Repraesentiert eine Kasse fuer Einkaeufe und erstellt Rechnungen.
 *
 * Die Klasse Kasse ermoeglicht es, eine Rechnung fuer die Einkaeufe eines
 * Kunden zu erstellen. Sie verwendet Informationen ueber den Kunden und das
 * Lager, um die Rechnung zu generieren.
 */
class Kasse {
public:
  /**
   * @brief Konstruktor fuer die Kasse mit einem Kunden und einem Lager.
   *
   * @param kunde Eine Konstante Referenz auf einen Kunden, dessen Einkaeufe
   * abgerechnet werden sollen.
   * @param lager Eine Referenz auf ein Lager, das fuer die Rechnung benoetigt
   * wird.
   */
  Kasse(Kunde const &kunde, Lager &lager);

  /**
   * @brief Erstellt die Rechnung fuer die Einkaeufe und gibt sie auf den
   * angegebenen Ausgabestrom aus.
   *
   * @param os Der Ausgabestrom, auf dem die Rechnung ausgegeben werden soll.
   */
  void rechnung(ostream &os);

  /**
   * @brief Gibt die Rechnungsdetails auf den angegebenen Ausgabestrom aus.
   *
   * @param os Der Ausgabestrom, auf dem die Rechnungsdetails ausgegeben werden
   * sollen.
   * @param date Das Datum der Rechnung.
   * @param rechnungsnummer Die Rechnungsnummer.
   * @param print_auswahl Gibt an, ob detaillierte Informationen zu den
   * ausgewaehlten Artikeln gedruckt werden sollen.
   */
  void printRechnung(ostream &os, const string &date,
                     const string &rechnungsnummer, bool print_auswahl);

private:
  Kunde const &kunde; ///< Konstante Referenz auf den Kunden fuer die Rechnung.
  Lager &lager;       ///< Referenz auf das Lager fuer die Rechnung.
};
```

```
70 #endif // !KASSE_HH
```

1.7 Datei 'Kasse.cc'

Listing 1.7 Die Header-Datei lager.cc

```
/**
 * @file laden.hh
 * @authors Yaman Alsaady, Oliver Schmidt
 * @brief Enthaelt die Deklaration der Klasse Kasse.
 * @version 0.3
 * @date 2023-11-13
 *
 * Dieses Header-Datei enthaelt die Definitionen von Klassen und
 * Funktionen zur Verwaltung von der Kasse in einem C++-Programm.
 *
 * @copyright Copyright (c) 2023
 *
 */

#include "kasse.hh"
#include "laden.hh"
#include "lager.hh"
#include <ctime>
#include <filesystem>
#include <fstream>
#include <iomanip>
#include <ios>
#include <iostream>
#include <ostream>
#include <string>
#include <unistd.h>
#include <vector>

using namespace std;

#define CLEAR u8"\033[2J\033[1;1H"
Kasse::Kasse(Kunde const &kunde, Lager &lager) : kunde(kunde), lager(lager) {}

void Kasse::rechnung(ostream &os) {

  time_t now = time(0);
  struct tm currentTime;
  localtime_r(&now, &currentTime);
  int year = currentTime.tm_year + 1900; // Jahr seit 1900
  int month = currentTime.tm_mon + 1;    // Monat von 0 bis 11
  int day = currentTime.tm_mday;         // Tag des Monats

  string date = to_string(year) + "-" + to_string(month) + "-" + to_string(day);
  string rechnungsnummer = to_string(month) + to_string(year) + to_string(day);

  filesystem::path currentDir = filesystem::current_path();
  string dateiname = "rechnungen/" + date + "_" + kunde.getName() + ".txt";

  if (!filesystem::exists("rechnungen")) {
    filesystem::create_directory("rechnungen");
  }
  ofstream datei(dateiname);

  cout << CLEAR;
  printRechnung(os, date, rechnungsnummer, true);
  string wahl;
  while (true) {
    cin >> wahl;
    if (wahl[0] == 'q') {
      os << "\n" << endl;
      break;
    }
    if (wahl[0] == 'p') {
      os << "\n" << endl;
      // string dateiname = kunde.getName() + ".txt";
      ofstream datei(dateiname);
      if (datei.is_open()) {
        printRechnung(datei, date, rechnungsnummer, false);
        datei.close();
```

```
70        }
71        cout << CLEAR;
72        cout << "Rechnung liegt hier: " << currentDir << "/" << dateiname << endl;
73        break;
74      }
75      cout << "Falsche Eingabe!" << endl;
76    }
77  }
78
79  void Kasse::printRechnung(ostream &os, const string &date,
80                            const string &rechnungsnummer, bool print_auswahl) {
81    string one_long = "-----------------------------------------------------";
82    string double_short = "-----------------------\n-----------------------";
83    double sum = 0;
84
85    os << "Rechnung des Warenwirtschaftssystems" << endl;
86    os << "Rechnungsnummer: " << rechnungsnummer << endl;
87    os << "Kunde: " << kunde.getName() << endl;
88    os << "Rechnungdatum: " << date << "\n" << endl;
89
90    for (Kunde::waren ware : kunde.getWarenkorb()) {
91      Artikel artikel = lager.getArtikel(ware.artikelnummer);
92      os << one_long << endl;
93      os << artikel.getName() << "\t" << artikel.getNormpreis() << " x ";
94      os << ware.menge << " " << artikel.getStrMasseinheit() << "/EUR";
95      os << "\t" << artikel.getNormpreis() * ware.menge << "EUR" << endl;
96      artikel.setLagerbestand(artikel.getLagerbestand() - ware.menge);
97      if (print_auswahl == false) {
98        lager.updateArtikel(ware.artikelnummer, new Artikel(artikel));
99      }
100     sum += artikel.getNormpreis() * ware.menge;
101   }
102
103   os << "\n"
104      << "Summe Netto:\t" << sum << "EUR" << endl;
105   os << "MwSt. 19%:\t" << sum * 0.19 << "EUR" << endl;
106   os << "Gesamt:\t\t" << sum * (1 - 0.19) << "EUR" << endl;
107   os << double_short << endl;
108
109   if (print_auswahl == true) {
110     os << "Beenden q:" << endl;
111     os << "Drucken p:" << endl;
112     os << "Auswahl: ";
113   }
114 }
```

1.8 Datei ’Makefile’

Listing 1.8 Die Header-Datei lager.cc

```
CXX = g++
CFLAGS = -Wall -Wextra -pedantic
SRC1 = $(wildcard *.cc)
SRC2 = $(wildcard *.cpp)
OBJ1 = $(patsubst %.cc, build/%.o, $(SRC1))
OBJ2 = $(patsubst %.cpp, build/%.o, $(SRC2))

build/main: $(OBJ1) $(OBJ2)
  $(CXX) $(CFLAGS) $(OBJ1) $(OBJ2) -o $@

build/%.o: %.cc
  @mkdir -p build
  ${CXX} ${CFLAGS} -c $< -o $@

build/%.o: %.cpp
  @mkdir -p build
  ${CXX} ${CFLAGS} -c $< -o $@

all: clean build/main

clean:
  rm -rf build

run:
  ./build/main -i waren.txt -o $(shell date +'%d.%m.%Y').txt
```

Kapitel 2

Hierarchie-Verzeichnis

2.1 Klassenhierarchie

Die Liste der Ableitungen ist -mit Einschränkungen- alphabetisch sortiert:

Kapitel 3

Klassen-Verzeichnis

3.1 Auflistung der Klassen

Hier folgt die Aufzählung aller Klassen, Strukturen, Varianten und Schnittstellen mit einer Kurzbeschreibung:

Kapitel 4

Datei-Verzeichnis

4.1 Auflistung der Dateien

Hier folgt die Aufzählung aller dokumentierten Dateien mit einer Kurzbeschreibung:

Kapitel 5

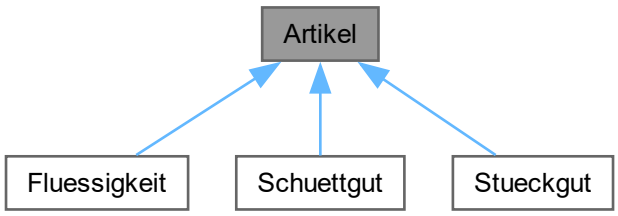
Klassen-Dokumentation

5.1 Artikel Klassenreferenz

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

`#include <lager.hh>`

Klassendiagramm für Artikel:



Öffentliche Methoden

- **Artikel** ()

  Standardkonstruktor fuer die Klasse "Artikel".
- Artikel (string name, string num, unsigned int bestand, masseinheit einheit, preis vp, preis np)

  Konstruktor fuer die Klasse "Artikel".
- ~**Artikel** ()

  Destruktor fuer die Klasse "Artikel".
- string getName () const

  Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam genutzt wird.
- string getArtikelnummer () const

  Gibt die Artikelnummer des Artikels zurueck.
- unsigned int getLagerbestand () const

Gibt den Lagerbestand des Artikels zurueck.

- masseinheit getMasseinheit () const

  Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurueck.
- string getStrMasseinheit () const

  Gibt die Masseinheit des Artikels als Zeichenkette zurueck.
- preis getVerkaufpreis () const

  Gibt den Verkaufspreis des Artikels zurueck.
- preis getNormpreis () const

  Gibt den Normalpreis des Artikels zurueck.
- int getGruppe () const

  Gibt die Warengruppe des Artikels zurueck.
- void setName (string name)

  Setzt den Namen des Artikels.
- void setArtikelnummer (string num)

  Setzt die Artikelnummer des Artikels.
- void setLagerbestand (unsigned int bestand)

  Setzt den Lagerbestand des Artikels.
- void setMasseinheit (masseinheit einheit)

  Setzt die Masseinheit des Artikels.
- void setVerkaufpreis (preis vp)

  Setzt den Verkaufspreis des Artikels.
- void setNormpreis (preis np)

  Setzt den Normalpreis des Artikels.
- ostream & print (ostream &outstream)

  Gibt die Artikelinformationen aus.

Geschützte Attribute

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

5.1.1 Ausführliche Beschreibung

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

5.1.2 Beschreibung der Konstruktoren und Destruktoren

5.1.2.1 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.

5.1.3 Dokumentation der Elementfunktionen

5.1.3.1 getArtikelnummer()

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

Gibt die Artikelnummer des Artikels zurueck.

Rückgabe

Die Artikelnummer des Artikels.

5.1.3.2 getGruppe()

```
int Artikel::getGruppe ( ) const
```

Gibt die Warengruppe des Artikels zurueck.

Rückgabe

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

5.1.3.3 getLagerbestand()

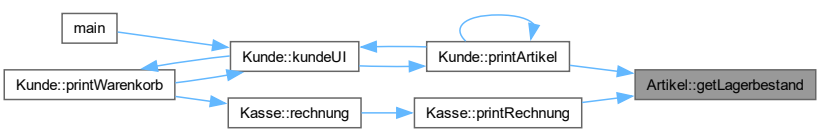
```
unsigned int Artikel::getLagerbestand ( ) const
```

Gibt den Lagerbestand des Artikels zurueck.

Rückgabe

Der Lagerbestand des Artikels.

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



5.1.3.4 getMasseinheit()

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

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

Rückgabe

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

5.1.3.5 getName()

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

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

Setzt die Warengruppe fuer Artikel.

Parameter

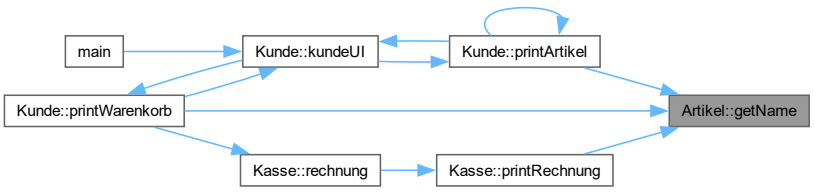
g	Die Warengruppe, die zugewiesen werden soll.

Gibt den Namen des Artikels zurueck.

Rückgabe

Der Name des Artikels.

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



5.1.3.6 getNormpreis()

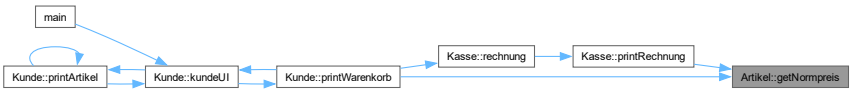
```
preis Artikel::getNormpreis ( ) const
```

Gibt den Normalpreis des Artikels zurueck.

Rückgabe

Der Normalpreis des Artikels.

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



5.1.3.7 getStrMasseinheit()

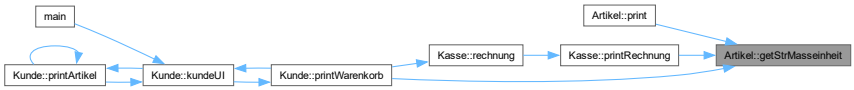
```
string Artikel::getStrMasseinheit ( ) const
```

Gibt die Masseinheit des Artikels als Zeichenkette zurueck.

Rückgabe

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

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



5.1.3.8 getVerkaufpreis()

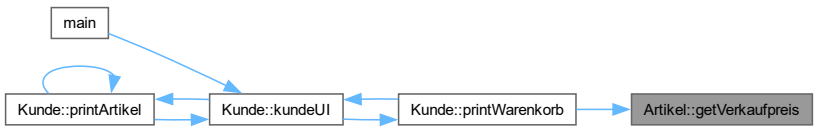
```
preis Artikel::getVerkaufpreis ( ) const
```

Gibt den Verkaufspreis des Artikels zurueck.

Rückgabe

Der Verkaufspreis des Artikels.

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



5.1.3.9 print()

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

Gibt die Artikelinformationen aus.

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

Parameter

os	Die Ausgabestromreferenz, in die die Informationen geschrieben werden.

Rückgabe

Die Ausgabestromreferenz, in die die Informationen geschrieben wurden.

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

5.1.3.10 setArtikelnummer()

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

Setzt die Artikelnummer des Artikels.

Parameter

num	Die neue Artikelnummer des Artikels.

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

5.1.3.11 setLagerbestand()

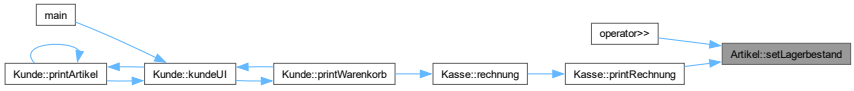
```
void Artikel::setLagerbestand (
            unsigned int bestand )
```

Setzt den Lagerbestand des Artikels.

Parameter

bestand	Der neue Lagerbestand des Artikels.

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



5.1.3.12 setMasseinheit()

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

Setzt die Masseinheit des Artikels.

Parameter

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

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

5.1.3.13 setName()

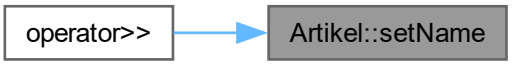
```
void Artikel::setName (
            string name )
```

Setzt den Namen des Artikels.

Parameter

name	Der neue Name des Artikels.

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



5.1.3.14 setNormpreis()

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

Setzt den Normalpreis des Artikels.

Parameter

np	Der neue Normalpreis des Artikels.

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

5.1.3.15 setVerkaufpreis()

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

Setzt den Verkaufspreis des Artikels.

Parameter

vp	Der neue Verkaufspreis des Artikels.

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

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

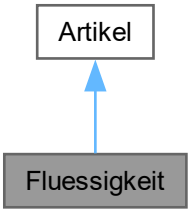
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.cc

5.2 Fluessigkeit Klassenreferenz

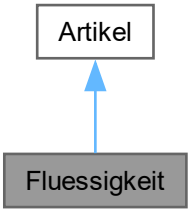
Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und spezialisiert sie fuer Fluessigkeits-Artikel.

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

Klassendiagramm für Fluessigkeit:



Zusammengehörigkeiten von Fluessigkeit:



Öffentliche Methoden

- Fluessigkeit (Artikel produkt)

  Konstruktor fuer die Klasse "Fluessigkeit" unter Verwendung eines bereits existierenden Artikels.
- 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** ()

  Standardkonstruktor fuer die Klasse "Artikel".
- Artikel (string name, string num, unsigned int bestand, masseinheit einheit, preis vp, preis np)

  Konstruktor fuer die Klasse "Artikel".
- ~**Artikel** ()

Destruktor fuer die Klasse "Artikel".

- string getName () const

  Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam genutzt wird.
- string getArtikelnummer () const

  Gibt die Artikelnummer des Artikels zurueck.
- unsigned int getLagerbestand () const

  Gibt den Lagerbestand des Artikels zurueck.
- masseinheit getMasseinheit () const

  Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurueck.
- string getStrMasseinheit () const

  Gibt die Masseinheit des Artikels als Zeichenkette zurueck.
- preis getVerkaufpreis () const

  Gibt den Verkaufspreis des Artikels zurueck.
- preis getNormpreis () const

  Gibt den Normalpreis des Artikels zurueck.
- int getGruppe () const

  Gibt die Warengruppe des Artikels zurueck.
- void setName (string name)

  Setzt den Namen des Artikels.
- void setArtikelnummer (string num)

  Setzt die Artikelnummer des Artikels.
- void setLagerbestand (unsigned int bestand)

  Setzt den Lagerbestand des Artikels.
- void setMasseinheit (masseinheit einheit)

  Setzt die Masseinheit des Artikels.
- void setVerkaufpreis (preis vp)

  Setzt den Verkaufspreis des Artikels.
- void setNormpreis (preis np)

  Setzt den Normalpreis des Artikels.
- ostream & print (ostream &outstream)

  Gibt die Artikelinformationen aus.

Private Attribute

- double **volume**

Weitere Geerbte Elemente

Geschützte Attribute geerbt von Artikel

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

5.2.1 Ausführliche Beschreibung

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

Die Klasse "Fluessigkeit" erweitert die Basisfunktionalitaet der Klasse "Artikel" um die Beruecksichtigung des Volumens.

5.2.2 Beschreibung der Konstruktoren und Destruktoren

5.2.2.1 Fluessigkeit() [1/2]

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

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

Parameter

produkt	Der Artikel, aus dem ein Fluessigkeits-Artikel erstellt wird.

5.2.2.2 Fluessigkeit() [2/2]

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

Konstruktor fuer die Klasse "Fluessigkeit".

Parameter

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).

5.2.3 Dokumentation der Elementfunktionen

5.2.3.1 getVolume()

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

Gibt das Volumen des Fluessigkeits-Artikels zurueck.

Rückgabe

Das Volumen des Artikels.

5.2.3.2 setVerkaufpreis()

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

Parameter

vp	Der Verkaufspreis, der gesetzt werden soll.

5.2.3.3 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:

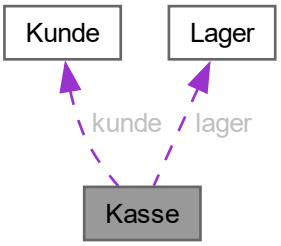
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.cc

5.3 Kasse Klassenreferenz

Repraesentiert eine Kasse fuer Einkaeufe und erstellt Rechnungen.

`#include <kasse.hh>`

Zusammengehörigkeiten von Kasse:



Öffentliche Methoden

- Kasse (Kunde const &kunde, Lager &lager)

  Konstruktor fuer die Kasse mit einem Kunden und einem Lager.
- void rechnung (ostream &os)

  Erstellt die Rechnung fuer die Einkaeufe und gibt sie auf den angegebenen Ausgabestrom aus.
- void printRechnung (ostream &os, const string &date, const string &rechnungsnummer, bool print_auswahl)

  Gibt die Rechnungsdetails auf den angegebenen Ausgabestrom aus.

Private Attribute

- Kunde const & **kunde**

  Konstante Referenz auf den Kunden fuer die Rechnung.
- Lager & **lager**

  Referenz auf das Lager fuer die Rechnung.

5.3.1 Ausführliche Beschreibung

Repraesentiert eine Kasse fuer Einkaeufe und erstellt Rechnungen.

Die Klasse Kasse ermoeglicht es, eine Rechnung fuer die Einkaeufe eines Kunden zu erstellen. Sie verwendet Informationen ueber den Kunden und das Lager, um die Rechnung zu generieren.

5.3.2 Beschreibung der Konstruktoren und Destruktoren

5.3.2.1 Kasse()

```
Kasse::Kasse (
            Kunde const & kunde,
            Lager & lager )
```

Konstruktor fuer die Kasse mit einem Kunden und einem Lager.

Parameter

kunde	Eine Konstante Referenz auf einen Kunden, dessen Einkaeufe abgerechnet werden sollen.
lager	Eine Referenz auf ein Lager, das fuer die Rechnung benoetigt wird.

5.3.3 Dokumentation der Elementfunktionen

5.3.3.1 printRechnung()

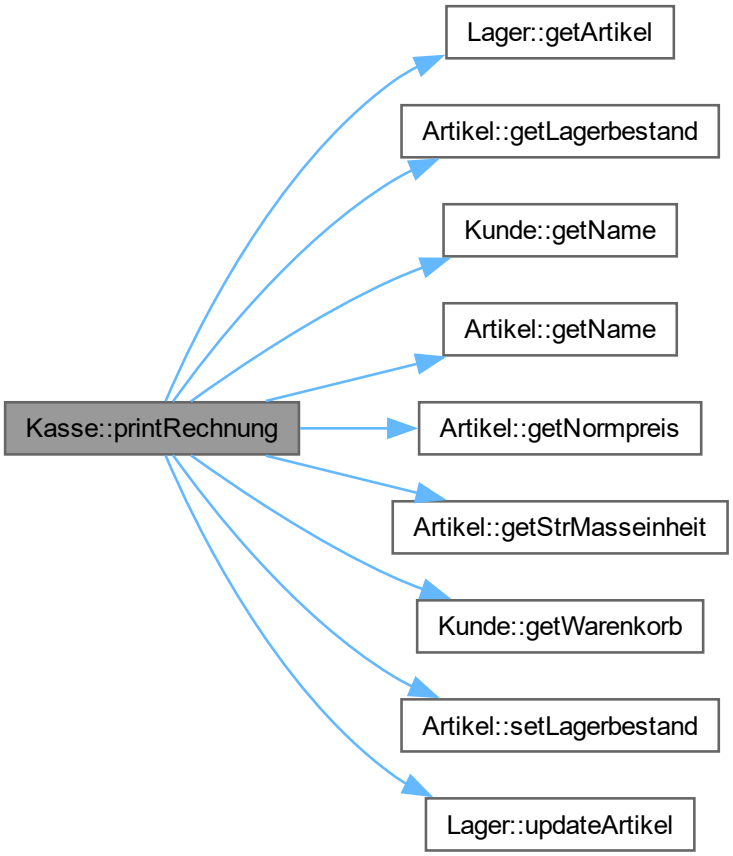
```
void Kasse::printRechnung (
            ostream & os,
            const string & date,
            const string & rechnungsnummer,
            bool print_auswahl )
```

Gibt die Rechnungsdetails auf den angegebenen Ausgabestrom aus.

Parameter

os	Der Ausgabestrom, auf dem die Rechnungsdetails ausgegeben werden sollen.
date	Das Datum der Rechnung.
rechnungsnummer	Die Rechnungsnummer.
print_auswahl	Gibt an, ob detaillierte Informationen zu den ausgewaehlten Artikeln gedruckt werden sollen.

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



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

5.3.3.2 rechnung()

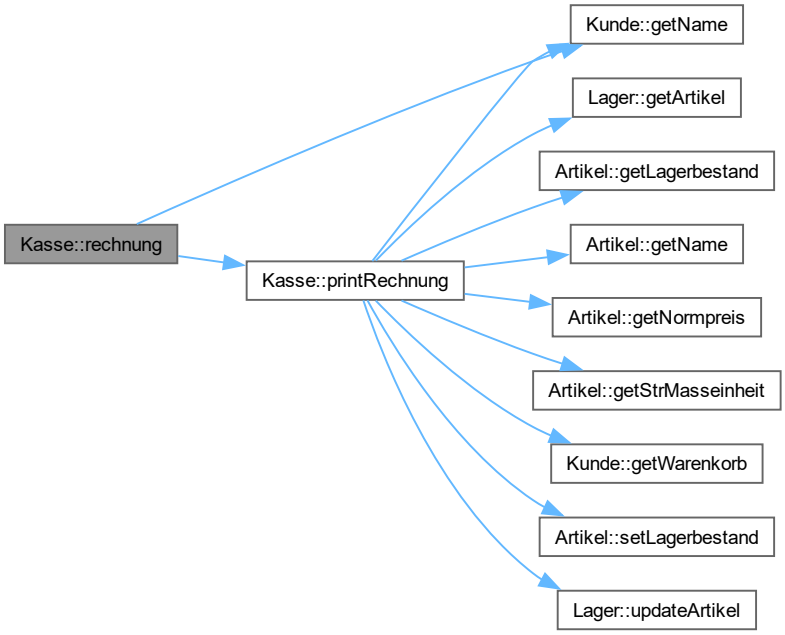
```
void Kasse::rechnung (
            ostream & os )
```

Erstellt die Rechnung fuer die Einkaeufe und gibt sie auf den angegebenen Ausgabestrom aus.

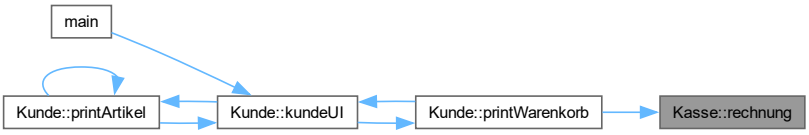
Parameter

os	Der Ausgabestrom, auf dem die Rechnung ausgegeben werden soll.

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



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



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/kasse.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/kasse.cc

5.4 Kunde Klassenreferenz

Repraesentiert einen Kunden mit Einkaufsfunktionen.

`#include <laden.hh>`

Klassen

- struct waren

  Struktur zur Darstellung von Waren im Warenkorb.

Öffentliche Methoden

- Kunde (string name, vector< Regal > const &regale)

  Konstruktor fuer einen Kunden mit einem Namen und einer Liste von Regalen.
- void **kundeUI** ()

  Oeffnet die Benutzeroberflaeche des Kunden fuer Einkaufsaktionen.
- string getName () const

  Gibt den Namen des Kunden zurueck.
- void printArtikel (int num)

  Gibt die Artikel in einem bestimmten Regal aus.
- void **printWarenkorb** ()

  Gibt den aktuellen Warenkorb des Kunden aus.
- vector< waren > getWarenkorb () const

  Gibt den aktuellen Warenkorb des Kunden zurueck.

Private Attribute

- string **name**
- vector< Regal > const & **regale**
- vector< waren > **warenkorb**

5.4.1 Ausführliche Beschreibung

Repraesentiert einen Kunden mit Einkaufsfunktionen.

Die Klasse Kunde stellt einen Kunden dar, der Einkaufsaktionen in einem Lager durchfuehren kann. Ein Kunde hat einen Namen, eine Liste von Regalen, die er durchsuchen kann, und einen Warenkorb, um Artikel hinzuzufuegen.

5.4.2 Beschreibung der Konstruktoren und Destruktoren

5.4.2.1 Kunde()

```
Kunde::Kunde (
            string name,
            vector< Regal > const & regale )
```

Konstruktor fuer einen Kunden mit einem Namen und einer Liste von Regalen.

Parameter

name	Der Name des Kunden.
regale	Eine Referenz auf eine Liste von Regalen, die der Kunde durchsuchen kann.

5.4.3 Dokumentation der Elementfunktionen

5.4.3.1 getName()

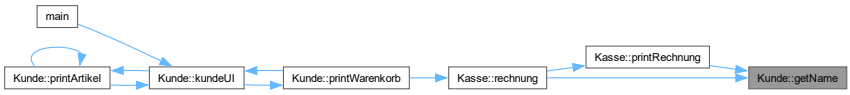
```
string Kunde::getName ( ) const
```

Gibt den Namen des Kunden zurueck.

Rückgabe

Der Name des Kunden.

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



5.4.3.2 getWarenkorb()

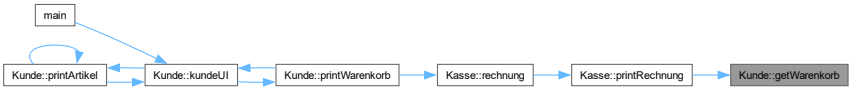
```
vector< Kunde::waren > Kunde::getWarenkorb ( ) const
```

Gibt den aktuellen Warenkorb des Kunden zurueck.

Rückgabe

Ein Vektor von Waren im Warenkorb.

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



5.4.3.3 printArtikel()

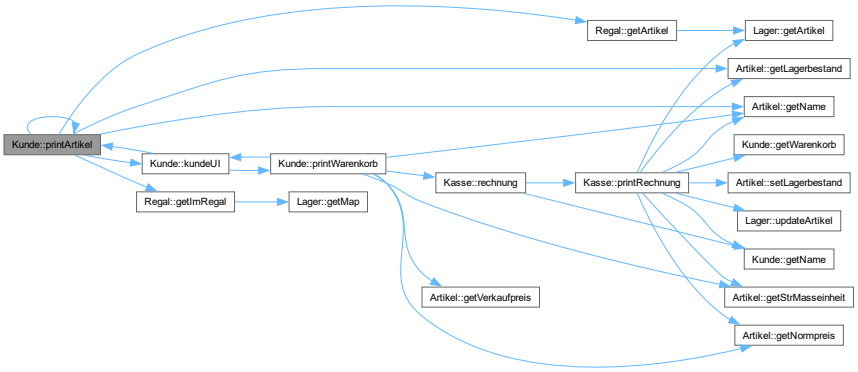
```
void Kunde::printArtikel (
            int num )
```

Gibt die Artikel in einem bestimmten Regal aus.

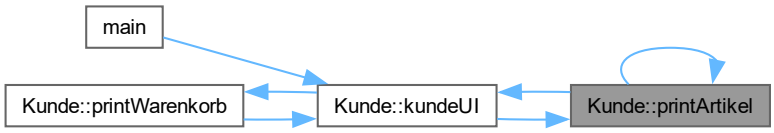
Parameter

num	Die Nummer des Regals, das durchsucht werden soll.

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



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



Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/laden.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/laden.cc

5.5 Lager Klassenreferenz

Klasse, die ein Lagerverwaltungssystem repraesentiert.

`#include <lager.hh>`

Öffentliche Typen

- typedef map< string, Artikel * > **artikelMap**

  Typdefinition fuer eine Map von Artikelnummern zu Artikeln.

Öffentliche Methoden

- **Lager** ()=default

  Standardkonstruktor fuer die Klasse Lager.
- ∼Lager ()

  Destruktor fuer die Klasse Lager.
- void readFile (string filename)

  Liest Artikelinformationen aus einer Datei und fuegt sie dem Lager hinzu.
- void write (ostream &os)

  Schreibt die Artikelinformationen in den angegebenen Ausgabeostream.
- void write (string filename)

  Schreibt die Artikelinformationen in eine Datei.
- Artikel getArtikel (string artikelnummer) const

  Gibt den Artikel mit der angegebenen Artikelnummer zurueck.
- artikelMap getMap ()

  Gibt die gesamte Map von Artikelnummern zu Artikeln zurueck.
- void updateArtikel (string num, Artikel ∗artikel)

  Aktualisiert die Informationen fuer einen Artikel in der Map.

Private Attribute

- artikelMap **lagerMap**

  < *Die Map von Artikelnummern zu Artikeln im Lager.*

5.5.1 Ausführliche Beschreibung

Klasse, die ein Lagerverwaltungssystem repraesentiert.

5.5.2 Beschreibung der Konstruktoren und Destruktoren

5.5.2.1 ∼Lager()

```
Lager::~Lager ( )
```

Destruktor fuer die Klasse Lager.

Der Destruktor durchlaeuft die Lager-Map und gibt den zugewiesenen Speicher fuer jeden Artikel frei, bevor das Lager-Objekt zerstoert wird.

5.5.3 Dokumentation der Elementfunktionen

5.5.3.1 getArtikel()

```
Artikel Lager::getArtikel (
            string artikelnummer ) const
```

Gibt den Artikel mit der angegebenen Artikelnummer zurueck.

Parameter

artikelnummer	Die Artikelnummer des gesuchten Artikels.

Rückgabe

Der Artikel mit der angegebenen Artikelnummer.

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

5.5.3.2 getMap()

```
Lager::artikelMap Lager::getMap ( )
```

Gibt die gesamte Map von Artikelnummern zu Artikeln zurueck.

Rückgabe

Die Map von Artikelnummern zu Artikeln.

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

5.5.3.3 readFile()

```
void Lager::readFile (
            string filename )
```

Liest Artikelinformationen aus einer Datei und fuegt sie dem Lager hinzu.

Parameter

filename	Der Dateiname der Eingabedatei.

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

5.5.3.4 updateArtikel()

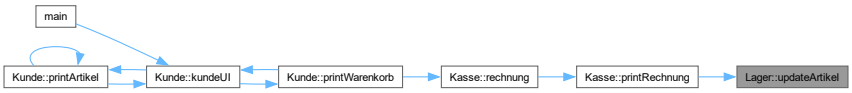
```
void Lager::updateArtikel (
            string num,
            Artikel * artikel )
```

Aktualisiert die Informationen fuer einen Artikel in der Map.

Parameter

num	Die Artikelnummer des zu aktualisierenden Artikels.
artikel	Der aktualisierte Artikel.

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



5.5.3.5 write() [1/2]

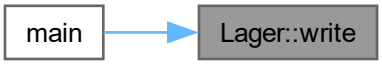
```
void Lager::write (
            ostream & os )
```

Schreibt die Artikelinformationen in den angegebenen Ausgabeostream.

Parameter

os	Der Ausgabeostream, in den die Informationen geschrieben werden.

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



5.5.3.6 write() [2/2]

```
void Lager::write (
            string filename )
```

Schreibt die Artikelinformationen in eine Datei.

Parameter

filename	Der Dateiname der Ausgabedatei.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.cc

5.6 Regal Klassenreferenz

Repraesentiert ein Regal im Lager.

```
#include <laden.hh>
```

Zusammengehörigkeiten von Regal:

Öffentliche Methoden

- Regal (string name, Lager &lager, int warengruppe)

  Konstruktor fuer ein Regal mit einer einzelnen Warengruppe.
- Regal (string name, Lager &lager, std::set< int > warengruppen)

  Konstruktor fuer ein Regal mit mehreren Warengruppen.
- Lager & getLager ()

  Gibt eine Referenz auf das Lager zurueck, zu dem das Regal gehoert.
- string getName () const

  Gibt den Namen des Regals zurueck.
- set< int > getWaren () const

  Gibt die Menge der Warengruppen zurueck, die dem Regal zugeordnet sind.
- Artikel getArtikel (string num) const

  Gibt einen Artikel im Regal anhand der Artikelnummer zurueck.
- vector< string > getImRegal () const

  Gibt eine Liste der Artikel im Regal zurueck.

Private Attribute

- string regalname

  < Der Name des Regals.
- Lager & **lager**

  Die Menge der Warengruppen, die dem Regal zugeordnet sind.
- std::set< int > **waren**

Freundbeziehungen

- ostream & operator<< (ostream &os, Regal regal)

  ueberschriebener Ausgabeoperator fuer die Klasse Regal.

5.6.1 Ausführliche Beschreibung

Repraesentiert ein Regal im Lager.

Die Klasse Regal stellt Informationen ueber ein Regal im Lager zur Verfuegung, einschliesslich des Regalnamens, des zugeordneten Lagers, der zugehoerigen Warengruppen und der im Regal befindlichen Artikel.

5.6.2 Beschreibung der Konstruktoren und Destruktoren

5.6.2.1 Regal() [1/2]

```
Regal::Regal (
            string name,
            Lager & lager,
            int warengruppe )
```

Konstruktor fuer ein Regal mit einer einzelnen Warengruppe.

Parameter

name	Der Name des Regals.
lager	Referenz auf das Lager, zu dem das Regal gehoert.
warengruppe	Die Warengruppe, die dem Regal zugeordnet ist.

5.6.2.2 Regal() [2/2]

```
Regal::Regal (
            string name,
            Lager & lager,
            std::set< int > warengruppen )
```

Konstruktor fuer ein Regal mit mehreren Warengruppen.

Parameter

name	Der Name des Regals.
lager	Referenz auf das Lager, zu dem das Regal gehoert.
warengruppen	Die Menge der Warengruppen, die dem Regal zugeordnet sind.

5.6.3 Dokumentation der Elementfunktionen

5.6.3.1 getArtikel()

```
Artikel Regal::getArtikel (
            string num ) const
```

Gibt einen Artikel im Regal anhand der Artikelnummer zurueck.

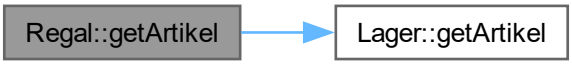
Parameter

num	Die Artikelnummer des gesuchten Artikels.

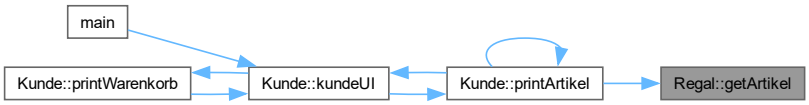
Rückgabe

Der gefundene Artikel im Regal.

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



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



5.6.3.2 getImRegal()

```
vector< string > Regal::getImRegal ( ) const
```

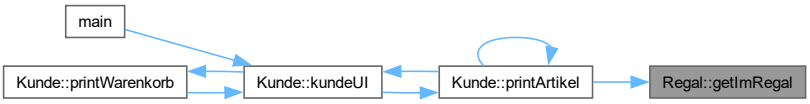
Gibt eine Liste der Artikel im Regal zurueck.

Rückgabe

Ein Vektor von Artikelnummern im Regal.

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

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



5.6.3.3 getLager()

```
Lager & Regal::getLager ( )
```

Gibt eine Referenz auf das Lager zurueck, zu dem das Regal gehoert.

Rückgabe

Eine Referenz auf das Lager.

5.6.3.4 getName()

```
string Regal::getName ( ) const
```

Gibt den Namen des Regals zurueck.

Rückgabe

Der Name des Regals.

5.6.3.5 getWaren()

```
std::set< int > Regal::getWaren ( ) const
```

Gibt die Menge der Warengruppen zurueck, die dem Regal zugeordnet sind.

Rückgabe

Eine Menge von Warengruppen.

5.6.4 Freundbeziehungen und Funktionsdokumentation

5.6.4.1 operator<<

```
ostream & operator<< (
            ostream & os,
            Regal regal )  [friend]
```

ueberschriebener Ausgabeoperator fuer die Klasse Regal.

Parameter

os	Der Ausgabestrom.
regal	Das Regal, das ausgegeben werden soll.

Rückgabe

Der Ausgabestrom.

5.6.5 Dokumentation der Datenelemente

5.6.5.1 regalname

```
string Regal::regalname  [private]
```

< Der Name des Regals.

Referenz auf das Lager, zu dem das Regal gehoert.

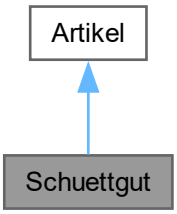
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/laden.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/laden.cc

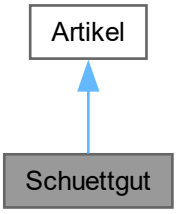
5.7 Schuettgut Klassenreferenz

`#include <lager.hh>`

Klassendiagramm für Schuettgut:



Zusammengehörigkeiten von Schuettgut:



Öffentliche Methoden

- Schuettgut (Artikel produkt)

  Konstruktor fuer die Klasse "Schuettgut" unter Verwendung eines bereits existierenden Artikels.
- Schuettgut (string name, string num, double groesse, preis np, unsigned int bestand=1)

  Konstruktor fuer die Klasse "Schuettgut".
- double getLosgroesse () const

  Gibt die Losgroesse des Schuettgut-Artikels zurueck.
- void setVerkaufpreis (preis vp)

  Setzt den Verkaufspreis des Schuettgut-Artikels basierend auf der Losgroesse.
- void setLosgroesse (double groesse)

  Setzt die Losgroesse des Schuettgut-Artikels.

Öffentliche Methoden geerbt von Artikel

- **Artikel** ()

  Standardkonstruktor fuer die Klasse "Artikel".
- Artikel (string name, string num, unsigned int bestand, masseinheit einheit, preis vp, preis np)

  Konstruktor fuer die Klasse "Artikel".
- ~**Artikel** ()

  Destruktor fuer die Klasse "Artikel".
- string getName () const

  Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam genutzt wird.
- string getArtikelnummer () const

  Gibt die Artikelnummer des Artikels zurueck.
- unsigned int getLagerbestand () const

  Gibt den Lagerbestand des Artikels zurueck.
- masseinheit getMasseinheit () const

  Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurueck.
- string getStrMasseinheit () const

  Gibt die Masseinheit des Artikels als Zeichenkette zurueck.
- preis getVerkaufpreis () const

  Gibt den Verkaufspreis des Artikels zurueck.
- preis getNormpreis () const

  Gibt den Normalpreis des Artikels zurueck.
- int getGruppe () const

  Gibt die Warengruppe des Artikels zurueck.
- void setName (string name)

  Setzt den Namen des Artikels.
- void setArtikelnummer (string num)

  Setzt die Artikelnummer des Artikels.
- void setLagerbestand (unsigned int bestand)

  Setzt den Lagerbestand des Artikels.
- void setMasseinheit (masseinheit einheit)

  Setzt die Masseinheit des Artikels.
- void setVerkaufpreis (preis vp)

  Setzt den Verkaufspreis des Artikels.
- void setNormpreis (preis np)

  Setzt den Normalpreis des Artikels.
- ostream & print (ostream &outstream)

  Gibt die Artikelinformationen aus.

Private Attribute

- double **losgroesse**

Weitere Geerbte Elemente

Geschützte Attribute geerbt von Artikel

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

5.7.1 Ausführliche Beschreibung

Die Klasse "Schuettgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Schuettgut-Artikel. Die Klasse "↩ Schuettgut" erweitert die Basisfunktionalitaet der Klasse "Artikel" um die Beruecksichtigung der Losgroesse.

5.7.2 Beschreibung der Konstruktoren und Destruktoren

5.7.2.1 Schuettgut() [1/2]

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

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

Parameter

produkt	Der Artikel, aus dem ein Schuettgut-Artikel erstellt wird.

5.7.2.2 Schuettgut() [2/2]

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

Konstruktor fuer die Klasse "Schuettgut".

Parameter

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).

5.7.3 Dokumentation der Elementfunktionen

5.7.3.1 getLosgroesse()

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

Gibt die Losgroesse des Schuettgut-Artikels zurueck.

Rückgabe

Die Losgroesse des Artikels.

5.7.3.2 setLosgroesse()

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

Setzt die Losgroesse des Schuettgut-Artikels.

Parameter

groesse	Die neue Losgroesse.

5.7.3.3 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:

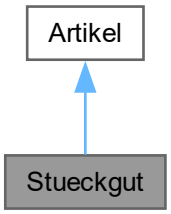
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.cc

5.8 Stueckgut Klassenreferenz

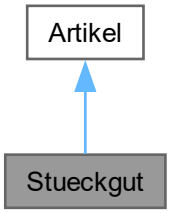
Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Stueckgut-Artikel.

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

Klassendiagramm für Stueckgut:



Zusammengehörigkeiten von Stueckgut:



Öffentliche Methoden

- Stueckgut (Artikel produkt)

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

Öffentliche Methoden geerbt von Artikel

- **Artikel** ()

  Standardkonstruktor fuer die Klasse "Artikel".
- Artikel (string name, string num, unsigned int bestand, masseinheit einheit, preis vp, preis np)

  Konstruktor fuer die Klasse "Artikel".
- ~**Artikel** ()

  Destruktor fuer die Klasse "Artikel".
- string getName () const

  Statische Warengruppen-Instanz, die fuer alle Artikel gemeinsam genutzt wird.
- string getArtikelnummer () const

  Gibt die Artikelnummer des Artikels zurueck.
- unsigned int getLagerbestand () const

  Gibt den Lagerbestand des Artikels zurueck.
- masseinheit getMasseinheit () const

  Gibt die Masseinheit des Artikels als Wert aus der Enumeration zurueck.
- string getStrMasseinheit () const

  Gibt die Masseinheit des Artikels als Zeichenkette zurueck.
- preis getVerkaufpreis () const

  Gibt den Verkaufspreis des Artikels zurueck.
- preis getNormpreis () const

  Gibt den Normalpreis des Artikels zurueck.
- int getGruppe () const

  Gibt die Warengruppe des Artikels zurueck.
- void setName (string name)

  Setzt den Namen des Artikels.
- void setArtikelnummer (string num)

  Setzt die Artikelnummer des Artikels.

- void setLagerbestand (unsigned int bestand)

  Setzt den Lagerbestand des Artikels.
- void setMasseinheit (masseinheit einheit)

  Setzt die Masseinheit des Artikels.
- void setVerkaufpreis (preis vp)

  Setzt den Verkaufspreis des Artikels.
- void setNormpreis (preis np)

  Setzt den Normalpreis des Artikels.
- ostream & print (ostream &outstream)

  Gibt die Artikelinformationen aus.

Weitere Geerbte Elemente

Geschützte Attribute geerbt von Artikel

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

5.8.1 Ausführliche Beschreibung

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

5.8.2 Beschreibung der Konstruktoren und Destruktoren

5.8.2.1 Stueckgut()

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

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:

- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.cc

5.9 Kunde::waren Strukturreferenz

Struktur zur Darstellung von Waren im Warenkorb.

`#include <laden.hh>`

Öffentliche Attribute

- string **artikelnummer**

  Die Artikelnummer.
- double **menge**

  Die Menge des Artikels im Warenkorb.

5.9.1 Ausführliche Beschreibung

Struktur zur Darstellung von Waren im Warenkorb.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/laden.hh

Kapitel 6

Datei-Dokumentation

6.1 kasse.hh

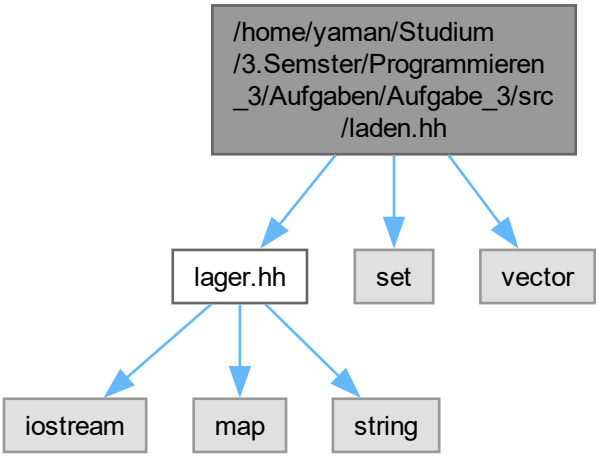
```
00001
00015 #ifndef KASSE_HH
00016 #define KASSE_HH
00017
00018 #include "laden.hh"
00019 #include "lager.hh"
00020 #include <iomanip>
00021 #include <string>
00022 #include <unistd.h>
00023 #include <vector>
00024
00033 class Kasse {
00034 public:
00043   Kasse(Kunde const &kunde, Lager &lager);
00044
00051   void rechnung(ostream &os);
00052
00063   void printRechnung(ostream &os, const string &date,
00064                      const string &rechnungsnummer, bool print_auswahl);
00065
00066 private:
00067   Kunde const &kunde;
00068   Lager &lager;
00069 };
00070 #endif // !KASSE_HH
```

6.2 /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/↩ Aufgabe_3/src/laden.hh-Dateireferenz

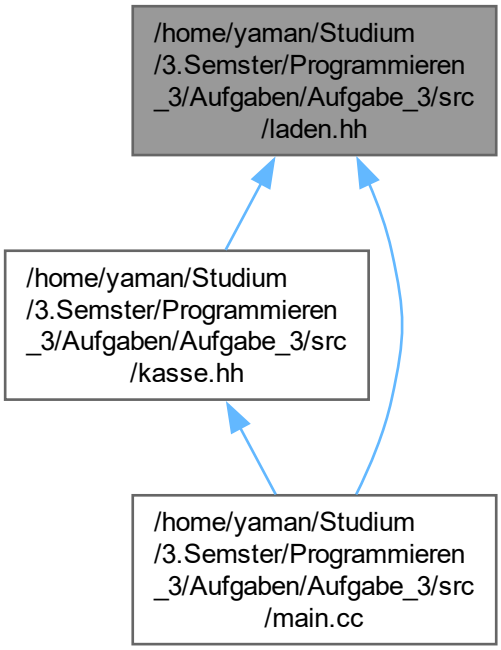
Enthaelt die Deklaration der Klasse Kasse.

```
#include "lager.hh"
#include <set>
#include <vector>
```

Include-Abhängigkeitsdiagramm für laden.hh:



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



Klassen

- class Regal

Repraesentiert ein Regal im Lager.

- class Kunde

  Repraesentiert einen Kunden mit Einkaufsfunktionen.

- struct Kunde::waren

  Struktur zur Darstellung von Waren im Warenkorb.

6.2.1 Ausführliche Beschreibung

Enthaelt die Deklaration der Klasse Kasse.

Enthaelt die Deklaration der Klasse Regal und der Klasse Kunde.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.3

Datum

2023-11-13

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

Copyright

Copyright (c) 2023

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.3

Datum

2023-11-13

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

Copyright

Copyright (c) 2023

6.3 laden.hh

gehe zur Dokumentation dieser Datei

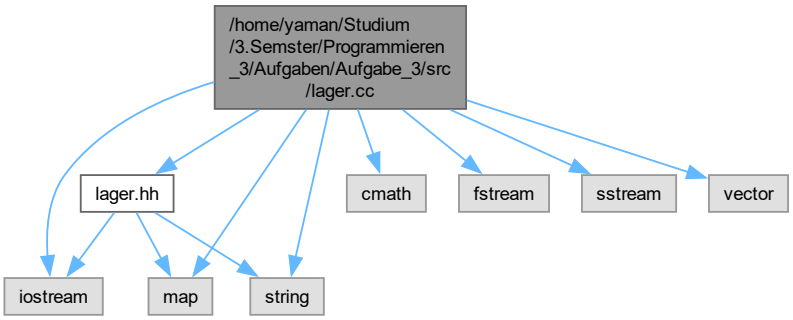
```
00001
00015 #ifndef LADEN_HH
00016 #define LADEN_HH
00017
00018 #include "lager.hh"
00019 #include <set>
00020 #include <vector>
00021
00030 class Regal {
00031 public:
00039   Regal(string name, Lager &lager, int warengruppe);
00040
00049   Regal(string name, Lager &lager, std::set<int> warengruppen);
00050
00056   Lager &getLager();
00057
00063   string getName() const;
00064
00071   set<int> getWaren() const;
00072
00079   Artikel getArtikel(string num) const;
00080
00086   vector<string> getImRegal() const;
00087
00095   friend ostream &operator«(ostream &os, Regal regal);
00096
00097 private:
00099   string regalname;
00101   Lager &lager;
00103   std::set<int> waren;
00104 };
00105
00114 class Kunde {
00115 public:
00124   Kunde(string name, vector<Regal> const &regale);
00125
00129   void kundeUI();
00130
00136   string getName() const;
00137
00143   void printArtikel(int num);
00144
00148   void printWarenkorb();
00149
00153   typedef struct {
00154     string artikelnummer;
00155     double menge;
00156   } waren;
00157
00163   vector<waren> getWarenkorb() const;
00164
00165 private:
00166   string name;
00167   vector<Regal> const &regale;
00168   vector<waren> warenkorb;
00169 };
00170
00171 #endif // !LADEN_HH
```

6.4 /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/↩ Aufgabe_3/src/lager.cc-Dateireferenz

Implementierung der Lagerverwaltungsfunktionen.

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

`#include <vector>`
Include-Abhängigkeitsdiagramm für lager.cc:



Funktionen

- static double **rounding** (double)
- std::ostream & **operator**<< (std::ostream &os, Artikel produkt)
- void operator>> (istream &is, Artikel &produkt)

  ueberladen des Eingabeoperators fuer die Artikelklasse.

6.4.1 Ausführliche Beschreibung

Implementierung der Lagerverwaltungsfunktionen.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.3

Datum

2023-11-13

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

Copyright

Copyright (c) 2023

6.4.2 Dokumentation der Funktionen

6.4.2.1 operator>>()

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

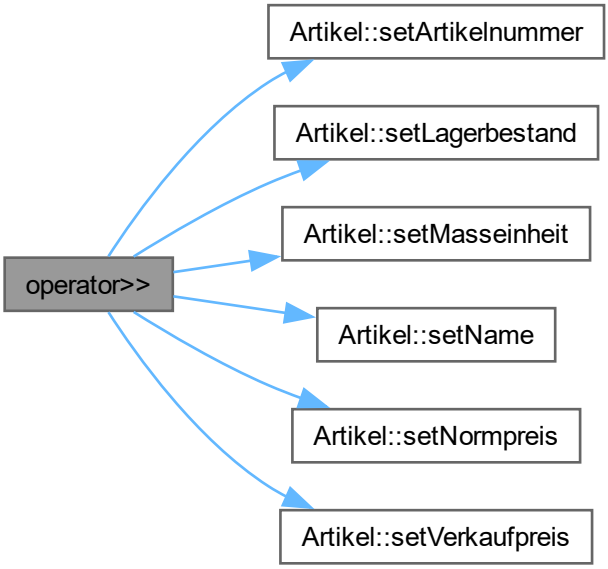
ueberladen des Eingabeoperators fuer die Artikelklasse.

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

Parameter

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

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

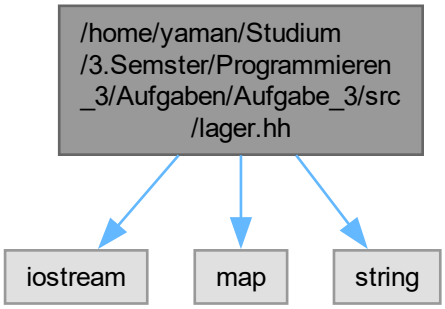


6.5 /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/↩ Aufgabe_3/src/lager.hh-Dateireferenz

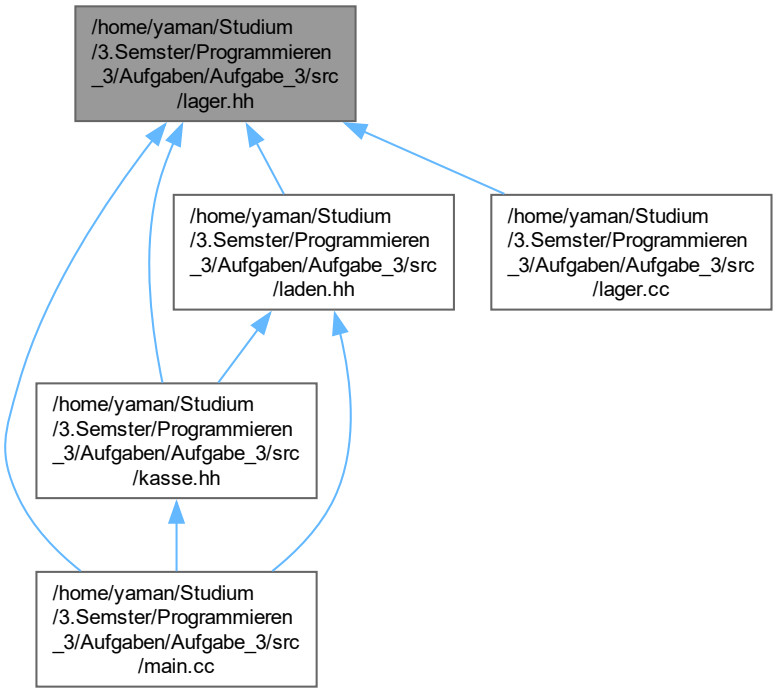
Definitionen der Lagerverwaltungsfunktionen.

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

Include-Abhängigkeitsdiagramm für lager.hh:



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



Klassen

- class 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.
- class Lager

  Klasse, die ein Lagerverwaltungssystem repraesentiert.

Typdefinitionen

- typedef double **preis**
- typedef int **Nummer**

Aufzählungen

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

Funktionen

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

  ueberladen des Ausgabeoperators fuer die Artikelklasse.
- void operator>> (istream &is, Artikel &produkt)

  ueberladen des Eingabeoperators fuer die Artikelklasse.

6.5.1 Ausführliche Beschreibung

Definitionen der Lagerverwaltungsfunktionen.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.3

Datum

2023-11-13

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

Copyright

Copyright (c) 2023

6.5.2 Dokumentation der Funktionen

6.5.2.1 operator<<()

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

ueberladen des Ausgabeoperators fuer die Artikelklasse.

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

Parameter

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

Rückgabe

Die Ausgabestromreferenz, in die die Informationen geschrieben wurden.

6.5.2.2 operator>>()

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

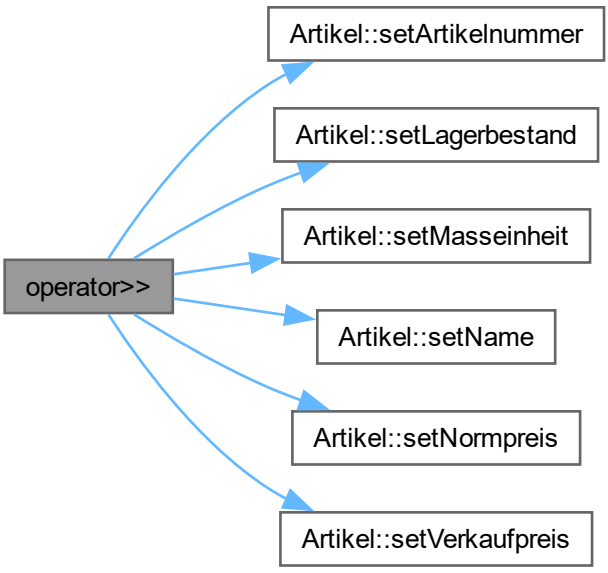
ueberladen des Eingabeoperators fuer die Artikelklasse.

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

Parameter

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

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



6.6 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 typedef int Nummer;
00026
00031 class Artikel {
00032 protected:
00033   string artikelname;
00034   string artikelnummer;
00035   unsigned int lagerbestand;
00036   masseinheit einheit;
00037   preis verkaufpreis;
00038   preis normpreis;
00039
00040 public:
00044   Artikel();
00045
00056   Artikel(string name, string num, unsigned int bestand, masseinheit einheit,
00057           preis vp, preis np);
00058
00059   // Getter-Funktionen
00060
00064   ~Artikel();
00065
00070   // static Warengruppen gruppe;
00071
00077   // static void setGruppe(Warengruppen g);
00078
00084   string getName() const;
00085
00091   string getArtikelnummer() const;
00092
00098   unsigned int getLagerbestand() const;
00099
00107   masseinheit getMasseinheit() const;
00108
00114   string getStrMasseinheit() const;
00115
00121   preis getVerkaufpreis() const;
00122
00128   preis getNormpreis() const;
00129
00136   int getGruppe() const;
00137
00138   // Setter-Funktionen
00139
00145   void setName(string name);
00146
00152   void setArtikelnummer(string num);
00153
00159   void setLagerbestand(unsigned int bestand);
00160
00166   void setMasseinheit(masseinheit einheit);
00167
00173   void setVerkaufpreis(preis vp);
00174
00180   void setNormpreis(preis np);
00181
00194   ostream &print(ostream &outstream);
00195 };
00209 ostream &operator«(ostream &os, Artikel produkt);
00210
00222 void operator»(istream &is, Artikel &produkt);
00223
00228 class Stueckgut : public Artikel {
00229 private:
00230 public:
00239   Stueckgut(Artikel produkt);
00240   Stueckgut(string name, string num, preis vp, unsigned int bestand = 1);
00241 };
00242
00249 class Schuettgut : public Artikel {
00250 private:
00251   double losgroesse;
```

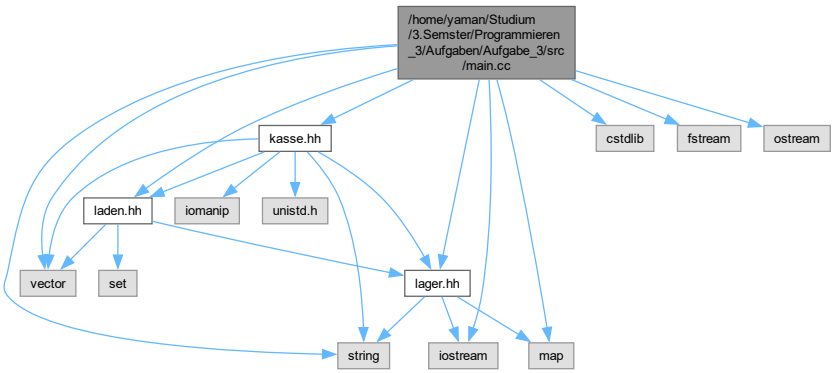
```
00252
00253 public:
00260   Schuettgut(Artikel produkt);
00261
00271   Schuettgut(string name, string num, double groesse, preis np,
00272              unsigned int bestand = 1);
00273
00279   double getLosgroesse() const;
00280
00287   void setVerkaufpreis(preis vp);
00288
00294   void setLosgroesse(double groesse);
00295 };
00296
00304 class Fluessigkeit : public Artikel {
00305 private:
00306   double volume;
00307
00308 public:
00316   Fluessigkeit(Artikel produkt);
00317
00328   Fluessigkeit(string name, string num, double vol, preis np,
00329                unsigned int bestand = 1);
00330
00336   double getVolume() const;
00337
00342   void setVerkaufpreis(preis vp);
00343
00349   void setVolume(double vol);
00350 };
00355 class Lager {
00356 public:
00360   Lager() = default;
00361
00368   ~Lager();
00369
00373   typedef map<string, Artikel *> artikelMap;
00374
00380   void readFile(string filename);
00381
00386   void write(ostream &os);
00387
00392   void write(string filename);
00393
00399   Artikel getArtikel(string artikelnummer) const;
00400
00405   artikelMap getMap();
00406
00412   void updateArtikel(string num, Artikel *artikel);
00413
00414 private:
00416   artikelMap lagerMap;
00417 };
00418 #endif // !LAGER_HH
```

6.7 /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/↩ Aufgabe_3/src/main.cc-Dateireferenz

Hauptprogramm fuer das Lagerverwaltungssystem.

```
#include "kasse.hh"
#include "laden.hh"
#include "lager.hh"
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <map>
#include <ostream>
#include <string>
#include <vector>
```

Include-Abhängigkeitsdiagramm für main.cc:



Makrodefinitionen

- #define **CLEAR** u8"\033[2J\033[1;1H"

  Definition fuer den Befehl zum Loeschen des Konsolenbildschirms.

Funktionen

- int main (int argc, char ∗argv[ ])

  Hauptfunktion des Programms.

6.7.1 Ausführliche Beschreibung

Hauptprogramm fuer das Lagerverwaltungssystem.

Autoren

Yaman Alsaady, Oliver Schmidt

Version

0.3

Datum

2023-11-13

Diese Datei dient als Einstiegspunkt fuer das Lagerverwaltungssystem. Sie liest Befehlszeilenargumente, initialisiert das Lager und die Regale, ermoeglicht dem Benutzer das Einkaufen und fuehrt die entsprechenden Operationen aus.

Copyright

6.7.2 Dokumentation der Funktionen

6.7.2.1 main()

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

Hauptfunktion des Programms.

Diese Funktion dient als Einstiegspunkt fuer das Lagerverwaltungssystem. Sie liest Befehlszeilenargumente, initialisiert das Lager und die Regale, ermoeglicht dem Benutzer das Einkaufen und fuehrt die entsprechenden Operationen aus.

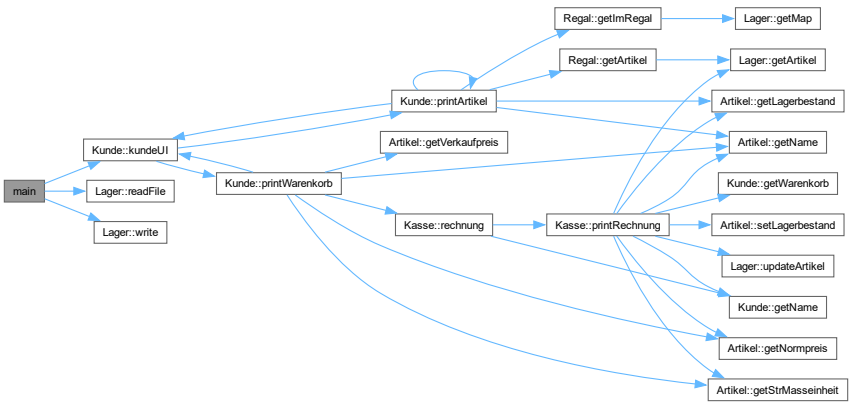
Parameter

argc	Die Anzahl der Befehlszeilenargumente.
argv	Ein Array von Zeichenketten, das die Befehlszeilenargumente enthaelt.

Rückgabe

Eine Ganzzahl, die den Programmstatus zurueckgibt (0 fuer Erfolg, andere Werte fuer Fehler).

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



Index