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, 12. Dezember 2023

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

Kapitel 1

Quellencode

1.1 Änderungen in laden.cc

Listing 1.1 Diff-Datei von laden.cc

```
diff --git a/Aufgabe_3/src/laden.cc b/Aufgabe_3/src/laden.cc
index 4ffb281..ad51d8a 100644
--- a/Aufgabe_3/src/laden.cc
+++ b/Aufgabe_3/src/laden.cc
@@ -38,21 +38,25 @@ std::set<int> Regal::getWaren() const { return waren; }

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

-vector<string> Regal::getImRegal() const {
+template <class OutputIterator>
+void Regal::products(OutputIterator out) 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;
+         *out = it->first;
+         out++;
        }
        it++;
      }
    }
+}
+vector<string> Regal::getImRegal() const {
+   vector<std::string> imRegal;
+   products(std::back_insert_iterator(imRegal));
    return imRegal;
 }
```

1.2 Änderungen in laden.hh

Listing 1.2 Diff-Datei von laden.hh

```
diff --git a/Aufgabe_3/src/laden.hh b/Aufgabe_3/src/laden.hh
index 07aa500..88e5853 100644
--- a/Aufgabe_3/src/laden.hh
+++ b/Aufgabe_3/src/laden.hh
@@ -48,6 +48,17 @@ public:
     */
    Regal(string name, Lager &lager, std::set<int> warengruppen);

+  /**
+   * @brief Template-Funktion zum Abrufen von Produkten im Regal
+   *
+   * Diese Funktion ruft Produkte aus dem Regal ab und gibt die Ergebnisse an den angegebenen
     Ausgabeiterator aus.
+   *
+   * @param out Der Ausgabeiterator, der das Ziel faer die Produkte
+   * repraesentiert.
+   */
+  template <class OutputIterator>
+  void products(OutputIterator out) const;
+
    /**
     * @brief Gibt eine Referenz auf das Lager zurueck, zu dem das Regal gehoert.
     *
```

1.3 Datei ’main.cc’

Listing 1.3 Die main.cc

```cpp
/**
 * @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 <iomanip>
#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);
  }
```

```
70
71    // Lager aus der Eingabedatei lesen
72    lager.readFile(fileread);
73
74    // Regale initialisieren
75    Regal gemueseRegal(R"(Gemuese)", lager, {40, 41});
76    Regal getraenkeRegal("Getraenke", lager, {43, 50, 55});
77    Regal sonderRegal("Sonderartikel", lager, 10);
78
79    regale.push_back(gemueseRegal);
80    regale.push_back(getraenkeRegal);
81    regale.push_back(sonderRegal);
82
83    // Hauptbenutzerschleife
84    while (true) {
85      cout << CLEAR;
86      cout << "Waehlen Sie aus!" << endl;
87      cout << setw(20) << left << "\tEinkaufen: "
88           << "n" << endl;
89      cout << setw(20) << left << "\tFeierabend: "
90           << "q" << endl;
91      cout << "\nAuswahl:";
92      cin >> wahl;
93
94      // Programm beenden, wenn 'q' ausgewaehlt wird
95      if (wahl == 'q') {
96        break;
97      }
98
99      // Einkaufen starten, wenn 'n' ausgewaehlt wird
100     if (wahl == 'n') {
101       cout << "Geben Ihre Name!" << endl;
102       cin >> vorname;
103       cin >> name;
104       Kunde kunde(vorname + string(" ") + name, regale);
105       kunde.kundeUI();
106       cout << CLEAR;
107     } else {
108       break;
109     }
110   }
111
112   // Lager aktualisieren und speichern
113   if (filewrite == "") {
114     lager.write("out.txt");
115   } else {
116     lager.write(filewrite);
117   }
118   return 0;
119 }
```

1.4 Datei ’Lager.hh’

Listing 1.4 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
```

1.5 Datei 'Lager.cc'

Listing 1.5 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;
}
```

1.6 Datei ’Laden.hh’

Listing 1.6 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 Template-Funktion zum Abrufen von Produkten im Regal
   *
   * Diese Funktion ruft Produkte aus dem Regal ab und gibt die Ergebnisse an den angegebenen
      Ausgabeiterator aus.
   *
   * @param out Der Ausgabeiterator, der das Ziel faer die Produkte
   * repraesentiert.
   */
  template <class OutputIterator>
  void products(OutputIterator out) const;

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

```

```
69    /**
70     * @brief Gibt den Namen des Regals zurueck.
71     *
72     * @return Der Name des Regals.
73     */
74    string getName() const;
75
76    /**
77     * @brief Gibt die Menge der Warengruppen zurueck, die dem Regal zugeordnet
78     * sind.
79     *
80     * @return Eine Menge von Warengruppen.
81     */
82    set<int> getWaren() const;
83
84    /**
85     * @brief Gibt einen Artikel im Regal anhand der Artikelnummer zurueck.
86     *
87     * @param num Die Artikelnummer des gesuchten Artikels.
88     * @return Der gefundene Artikel im Regal.
89     */
90    Artikel getArtikel(string num) const;
91
92    /**
93     * @brief Gibt eine Liste der Artikel im Regal zurueck.
94     *
95     * @return Ein Vektor von Artikelnummern im Regal.
96     */
97    vector<string> getImRegal() const;
98
99    /**
100    * @brief ueberschriebener Ausgabeoperator fuer die Klasse Regal.
101    *
102    * @param os Der Ausgabestrom.
103    * @param regal Das Regal, das ausgegeben werden soll.
104    * @return Der Ausgabestrom.
105    */
106   friend ostream &operator<<(ostream &os, Regal regal);
107
108 private:
109   ///< Der Name des Regals.
110   string regalname;
111   ///< Referenz auf das Lager, zu dem das Regal gehoert.
112   Lager &lager;
113   ///< Die Menge der Warengruppen, die dem Regal zugeordnet sind.
114   std::set<int> waren;
115 };
116
117 /**
118  * @class Kunde
119  * @brief Repraesentiert einen Kunden mit Einkaufsfunktionen.
120  *
121  * Die Klasse Kunde stellt einen Kunden dar, der Einkaufsaktionen in einem Lager
122  * durchfuehren kann. Ein Kunde hat einen Namen, eine Liste von Regalen, die er
123  * durchsuchen kann, und einen Warenkorb, um Artikel hinzuzufuegen.
124  */
125 class Kunde {
126 public:
127   /**
128    * @brief Konstruktor fuer einen Kunden mit einem Namen und einer Liste von
129    * Regalen.
130    *
131    * @param name Der Name des Kunden.
132    * @param regale Eine Referenz auf eine Liste von Regalen, die der Kunde
133    * durchsuchen kann.
134    */
135   Kunde(string name, vector<Regal> const &regale);
136
137   /**
138    * @brief Oeffnet die Benutzeroberflaeche des Kunden fuer Einkaufsaktionen.
139    */
140   void kundeUI();
141
```

```
142   /**
143    * @brief Gibt den Namen des Kunden zurueck.
144    *
145    * @return Der Name des Kunden.
146    */
147   string getName() const;
148 
149   /**
150    * @brief Gibt die Artikel in einem bestimmten Regal aus.
151    *
152    * @param num Die Nummer des Regals, das durchsucht werden soll.
153    */
154   void printArtikel(int num);
155 
156   /**
157    * @brief Gibt den aktuellen Warenkorb des Kunden aus.
158    */
159   void printWarenkorb();
160 
161   /**
162    * @brief Struktur zur Darstellung von Waren im Warenkorb.
163    */
164   typedef struct {
165     string artikelnummer; ///< Die Artikelnummer.
166     double menge;         ///< Die Menge des Artikels im Warenkorb.
167   } waren;
168 
169   /**
170    * @brief Gibt den aktuellen Warenkorb des Kunden zurueck.
171    *
172    * @return Ein Vektor von Waren im Warenkorb.
173    */
174   vector<waren> getWarenkorb() const;
175 
176 private:
177   string name;
178   vector<Regal> const &regale;
179   vector<waren> warenkorb;
180 };
181 
182 #endif // !LADEN_HH
```

1.7 Datei 'Laden.cc'

Listing 1.7 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 <iostream>
#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); }

template <class OutputIterator>
void Regal::products(OutputIterator out) const {
  Lager::artikelMap map = lager.getMap();
  for (int ware : waren) {
    Lager::artikelMap::iterator it = map.begin();
    while (it != map.end()) {
      int num = (*it->second).getGruppe();
      num /= 100;
      if (ware == num) {
        *out = it->first;
        out++;
      }
      it++;
    }
  }
}
vector<string> Regal::getImRegal() const {
  vector<std::string> imRegal;
  products(std::back_insert_iterator(imRegal));
  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;
```

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

```
143   double wahl2num;
144   cout << "Warenkorb: " << warenkorb.size() << endl;
145   cout << "Waehlen Sie einen Artikel aus\n" << left << endl;
146   cout << setw(15) << "";
147   cout << setw(30) << "Bezeichnung";
148   cout << setw(20) << "Lagerbestand";
149   cout << setw(20) << "Preis/Einheit" << endl;
150
151   cout << regale[num];
152   cout << setw(5) << "";
153   cout << "." << setw(9) << ":" << left;
154   cout << "Zurueck" << left;
155   cout << endl;
156   cout << setw(5) << "";
157   cout << "q" << setw(9) << ":" << left;
158   cout << "Beenden" << left;
159   cout << endl;
160   cout << endl;
161   while (true) {
162     cout << "Auswahl: ";
163     cin >> wahl1;
164     if (wahl1[0] == 'q') {
165       break;
166     }
167     if (wahl1[0] == '.') {
168       kundeUI();
169       break;
170     }
171     try {
172
173       wahl1Num = stoi(wahl1);
174       if (wahl1Num > Regal(regale[num]).getImRegal().size()) {
175       } else {
176         wahl1Num--;
177         string artnum = Regal(regale[num]).getImRegal()[wahl1Num];
178         Artikel artikel = Regal(regale[num]).getArtikel(artnum);
179         cout << artikel.getName() << endl;
180         cout << "Geben Sie die Menge" << endl;
181         cin >> wahl2;
182         wahl2num = stof(wahl2);
183         if (wahl2num <= artikel.getLagerbestand()) {
184           warenkorb.push_back({artnum, wahl2num});
185           cout << CLEAR;
186           float menge = warenkorb[warenkorb.size() - 1].menge;
187           cout << menge << " * " << artikel.getName() << endl;
188           sleep(1);
189           printArtikel(num);
190           break;
191         }
192       }
193     } catch (const std::exception &) {
194     }
195     cout << "Falsche Eingabe!" << endl;
196   }
197 }
198 void Kunde::printWarenkorb() {
199   string wahl;
200   int i = 0;
201   cout << CLEAR;
202   cout << "Warenkorb: " << warenkorb.size() << endl;
203   cout << "Waehlen Sie aus\n" << left << endl;
204   for (auto ware : warenkorb) {
205     Artikel artikel = regale[0].getArtikel(ware.artikelnummer);
206     i++;
207     cout.imbue(locale("de_DE.UTF-8"));
208     cout << setw(5) << "";
209     cout << i << setw(9) << ":" << left;
210     cout << setw(30) << artikel.getName();
211     cout << setw(6) << artikel.getVerkaufpreis() << "/" << setw(4)
212          << artikel.getVerkaufpreis() / artikel.getNormpreis();
213     cout << setw(20) << artikel.getStrMasseinheit();
214     cout << setw(20) << ware.menge;
215     cout << setw(20) << showbase << (artikel.getNormpreis() * ware.menge)
```

```
216           << endl;
217   }
218   cout << setw(5) << "";
219   cout << "k" << setw(9) << ":" << left;
220   cout << "Kasse" << left;
221   cout << endl;
222   cout << setw(5) << "";
223   cout << "." << setw(9) << ":" << left;
224   cout << "Zurueck" << left;
225   cout << endl;
226   cout << setw(5) << "";
227   cout << "q" << setw(9) << ":" << left;
228   cout << "Beenden" << left << endl;
229   cout << "Auswahl: ";
230   while (true) {
231     cin >> wahl;
232     if (wahl[0] == 'q') {
233       break;
234     }
235     if (wahl[0] == '.') {
236       kundeUI();
237       break;
238     }
239     if (wahl[0] == 'k') {
240       Kasse kasse(*this, Regal(regale[0]).getLager());
241       kasse.rechnung(cout);
242       break;
243     }
244     cout << "Falsche Eingabe!" << endl;
245   }
246 }
```

1.8 Datei 'Kasse.hh'

Listing 1.8 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
```

1.9 Datei 'Kasse.cc'

Listing 1.9 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 }
```

1.10 Datei ’Makefile’

Listing 1.10 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 Dateien mit einer Kurzbeschreibung:

Kapitel 5

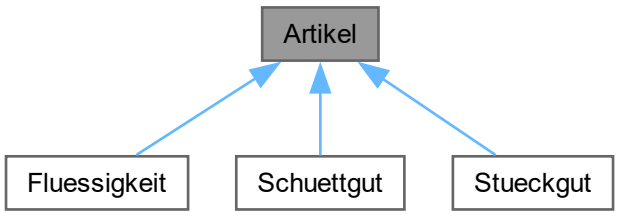
Klassen-Dokumentation

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.

Definiert in Zeile 31 der Datei lager.hh.

5.1.2 Beschreibung der Konstruktoren und Destruktoren

5.1.2.1 Artikel() [1/2]

```
Artikel::Artikel ( )
```

Standardkonstruktor fuer die Klasse "Artikel".

Definiert in Zeile 93 der Datei lager.cc.

5.1.2.2 Artikel() [2/2]

```
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.

Definiert in Zeile 94 der Datei lager.cc.

5.1.2.3 ~Artikel()

```
Artikel::~Artikel ( )
```

Destruktor fuer die Klasse "Artikel".

Definiert in Zeile 98 der Datei lager.cc.

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.

Definiert in Zeile 102 der Datei lager.cc.

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.

Definiert in Zeile 120 der Datei lager.cc.

5.1.3.3 getLagerbestand()

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

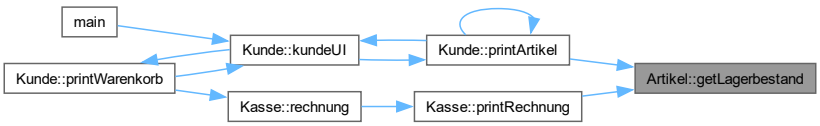
Gibt den Lagerbestand des Artikels zurueck.

Rückgabe

Der Lagerbestand des Artikels.

Definiert in Zeile 103 der Datei lager.cc.

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).

Definiert in Zeile 104 der Datei lager.cc.

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.

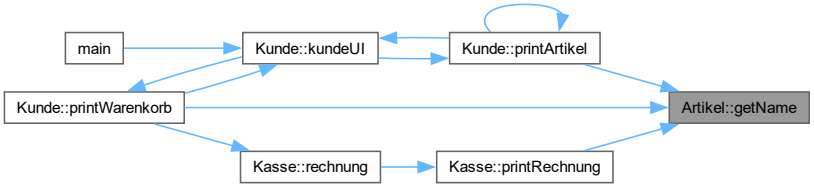
Gibt den Namen des Artikels zurueck.

Rückgabe

Der Name des Artikels.

Definiert in Zeile 101 der Datei lager.cc.

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



5.1.3.6 getNormpreis()

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

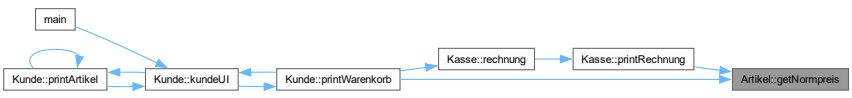
Gibt den Normalpreis des Artikels zurueck.

Rückgabe

Der Normalpreis des Artikels.

Definiert in Zeile 119 der Datei lager.cc.

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



5.1.3.7 getStrMasseinheit()

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

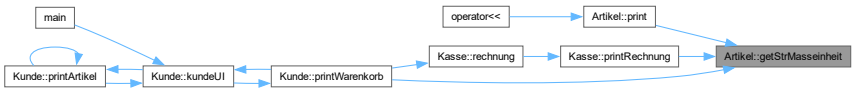
Gibt die Masseinheit des Artikels als Zeichenkette zurueck.

Rückgabe

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

Definiert in Zeile 105 der Datei lager.cc.

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



5.1.3.8 getVerkaufpreis()

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

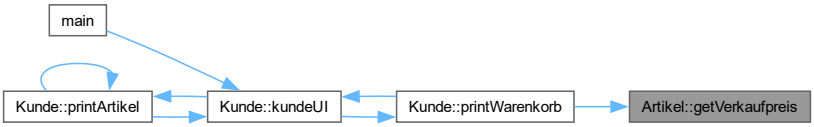
Gibt den Verkaufspreis des Artikels zurueck.

Rückgabe

Der Verkaufspreis des Artikels.

Definiert in Zeile 118 der Datei lager.cc.

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.

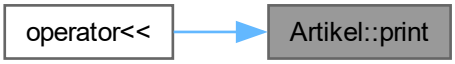
Rückgabe

Die Ausgabestromreferenz, in die die Informationen geschrieben wurden.

Definiert in Zeile 133 der Datei lager.cc.

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

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



5.1.3.10 setArtikelnummer()

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

Setzt die Artikelnummer des Artikels.

Parameter

num	Die neue Artikelnummer des Artikels.

Definiert in Zeile 127 der Datei lager.cc.

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.

Definiert in Zeile 128 der Datei lager.cc.

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).

Definiert in Zeile 129 der Datei lager.cc.

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

5.1.3.13 setName()

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

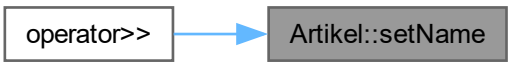
Setzt den Namen des Artikels.

Parameter

name	Der neue Name des Artikels.

Definiert in Zeile 126 der Datei lager.cc.

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.

Definiert in Zeile 131 der Datei lager.cc.

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.

Definiert in Zeile 130 der Datei lager.cc.

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

5.1.4 Dokumentation der Datenelemente

5.1.4.1 artikelname

```
string Artikel::artikelname  [protected]
```

Definiert in Zeile 33 der Datei lager.hh.

5.1.4.2 artikelnummer

```
string Artikel::artikelnummer  [protected]
```

Definiert in Zeile 34 der Datei lager.hh.

5.1.4.3 einheit

```
masseinheit Artikel::einheit  [protected]
```

Definiert in Zeile 36 der Datei lager.hh.

5.1.4.4 lagerbestand

```
unsigned int Artikel::lagerbestand  [protected]
```

Definiert in Zeile 35 der Datei lager.hh.

5.1.4.5 normpreis

```
preis Artikel::normpreis  [protected]
```

Definiert in Zeile 38 der Datei lager.hh.

5.1.4.6 verkaufpreis

```
preis Artikel::verkaufpreis  [protected]
```

Definiert in Zeile 37 der Datei lager.hh.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

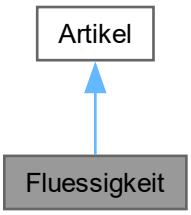
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.cc

5.2 Fluessigkeit Klassenreferenz

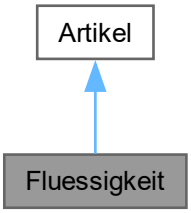
Die Klasse "Fluessigkeit" erbt von der Klasse "Artikel" und spezialisiert sie fuer Fluessigkeits-Artikel.

```
#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.

Definiert in Zeile 304 der Datei lager.hh.

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.

Definiert in Zeile 239 der Datei lager.cc.

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).

Definiert in Zeile 243 der Datei lager.cc.

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.

Definiert in Zeile 246 der Datei lager.cc.

5.2.3.2 setVerkaufpreis()

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

Parameter

vp	Der Verkaufspreis, der gesetzt werden soll.

Definiert in Zeile 251 der Datei lager.cc.

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

5.2.3.3 setVolume()

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

Setzt das Volumen des Fluessigkeits-Artikels.

Parameter

vol	Das neue Volumen.

Definiert in Zeile 247 der Datei lager.cc.

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

5.2.4 Dokumentation der Datenelemente

5.2.4.1 volume

```
double Fluessigkeit::volume  [private]
```

Definiert in Zeile 306 der Datei lager.hh.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

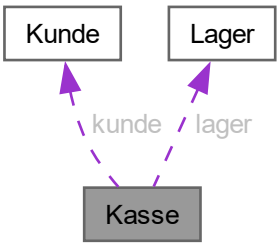
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.cc

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.

Definiert in Zeile 33 der Datei kasse.hh.

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.

Definiert in Zeile 32 der Datei kasse.cc.

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 )
```

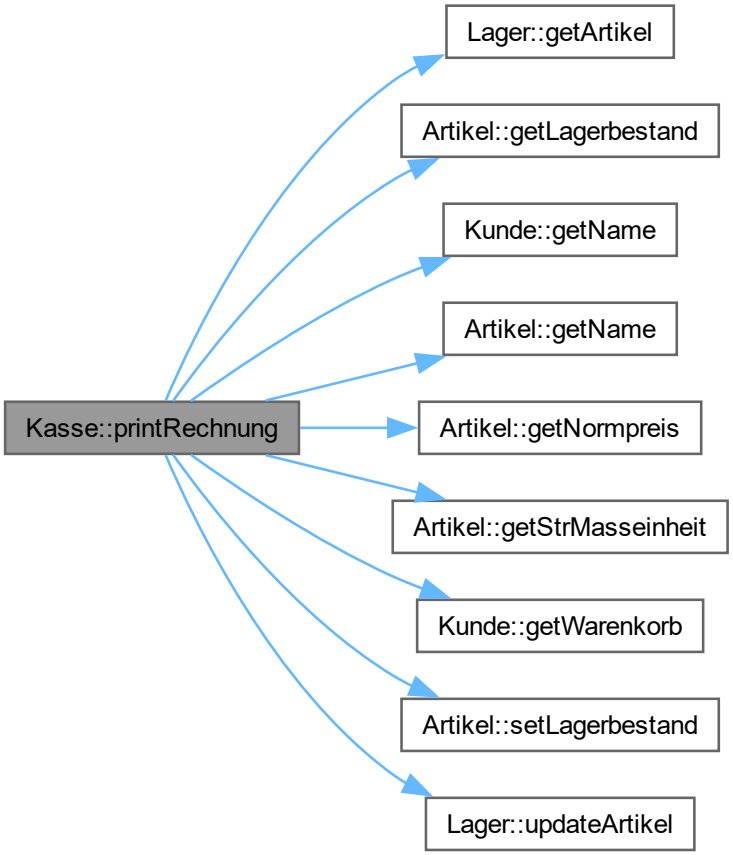
Gibt die Rechnungsdetails auf den angegebenen Ausgabestrom aus.

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.

Definiert in Zeile 79 der Datei kasse.cc.

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 )
```

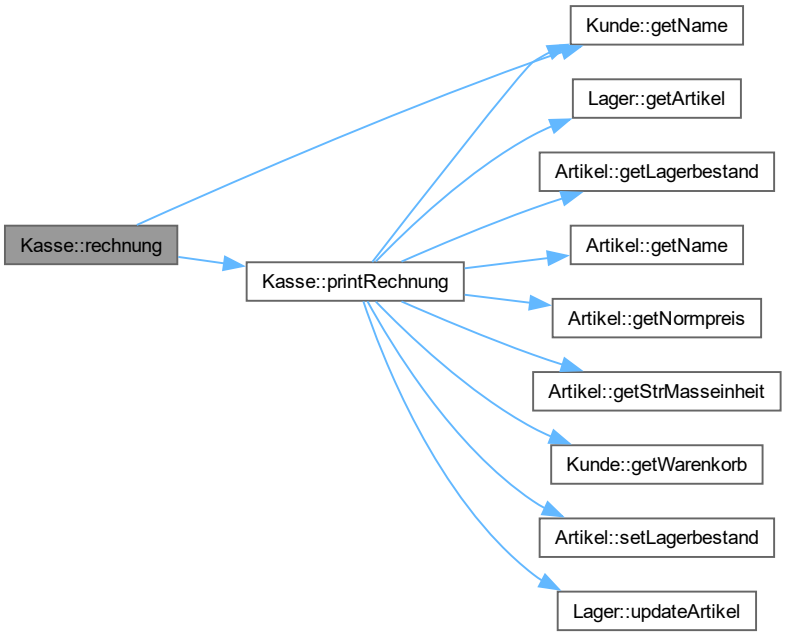
Erstellt die Rechnung fuer die Einkaeufe und gibt sie auf den angegebenen Ausgabestrom aus.

Parameter

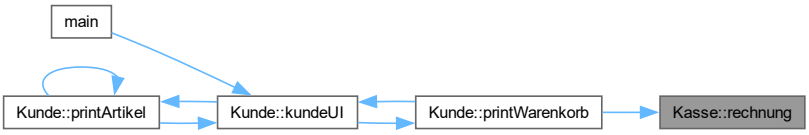
os	Der Ausgabestrom, auf dem die Rechnung ausgegeben werden soll.

Definiert in Zeile 34 der Datei kasse.cc.

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



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



5.3.4 Dokumentation der Datenelemente

5.3.4.1 kunde

```
Kunde const& Kasse::kunde  [private]
```

Konstante Referenz auf den Kunden fuer die Rechnung.

Definiert in Zeile 67 der Datei kasse.hh.

5.3.4.2 lager

```
Lager& Kasse::lager  [private]
```

Referenz auf das Lager fuer die Rechnung.

Definiert in Zeile 68 der Datei kasse.hh.

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.

Definiert in Zeile 125 der Datei laden.hh.

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.

Definiert in Zeile 83 der Datei laden.cc.

5.4.3 Dokumentation der Elementfunktionen

5.4.3.1 getName()

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

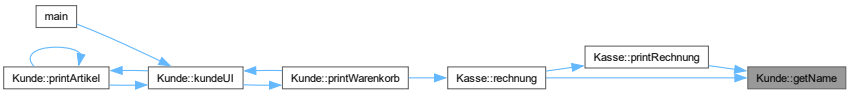
Gibt den Namen des Kunden zurueck.

Rückgabe

Der Name des Kunden.

Definiert in Zeile 88 der Datei laden.cc.

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



5.4.3.2 getWarenkorb()

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

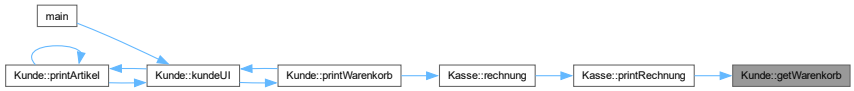
Gibt den aktuellen Warenkorb des Kunden zurueck.

Rückgabe

Ein Vektor von Waren im Warenkorb.

Definiert in Zeile 86 der Datei laden.cc.

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



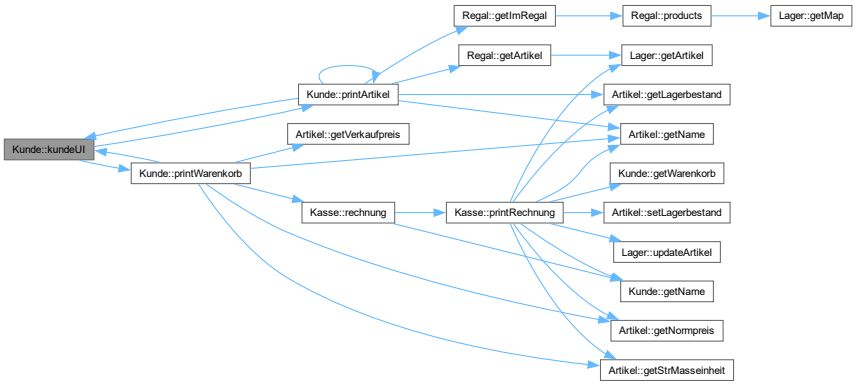
5.4.3.3 kundeUI()

```
void Kunde::kundeUI ( )
```

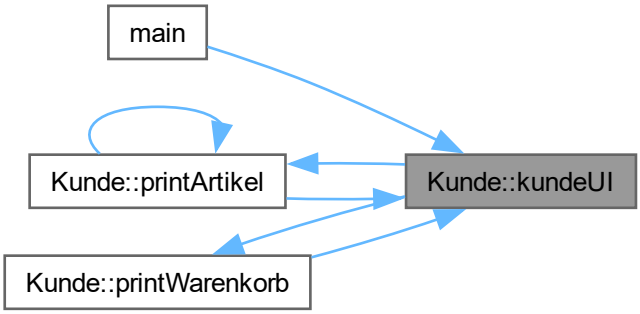
Oeffnet die Benutzeroberflaeche des Kunden fuer Einkaufsaktionen.

Definiert in Zeile 90 der Datei laden.cc.

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



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



5.4.3.4 printArtikel()

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

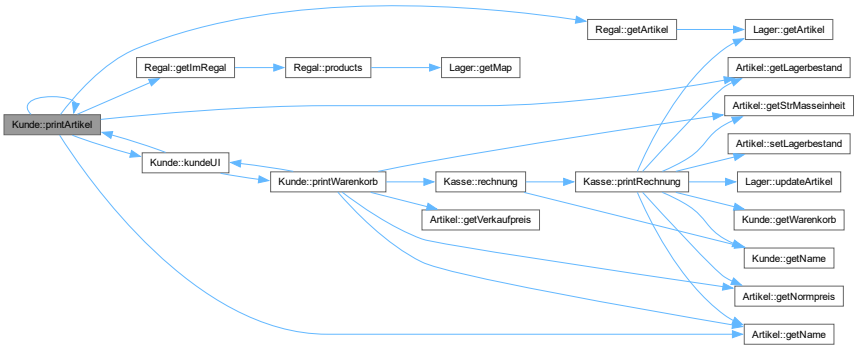
Gibt die Artikel in einem bestimmten Regal aus.

Parameter

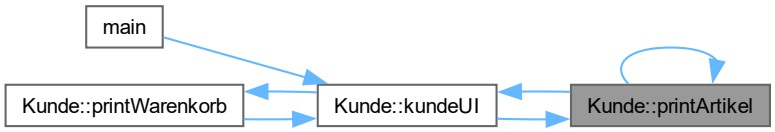
num	Die Nummer des Regals, das durchsucht werden soll.

Definiert in Zeile 139 der Datei laden.cc.

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



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



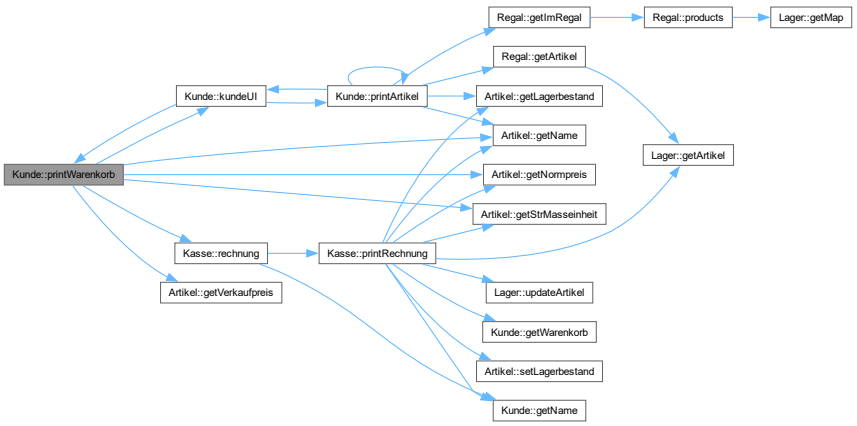
5.4.3.5 printWarenkorb()

```
void Kunde::printWarenkorb ( )
```

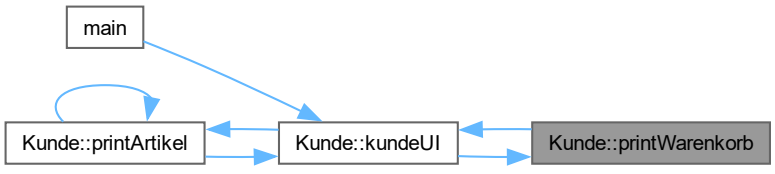
Gibt den aktuellen Warenkorb des Kunden aus.

Definiert in Zeile 198 der Datei laden.cc.

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



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



5.4.4 Dokumentation der Datenelemente

5.4.4.1 name

```
string Kunde::name  [private]
```

Definiert in Zeile 177 der Datei laden.hh.

5.4.4.2 regale

```
vector<Regal> const& Kunde::regale  [private]
```

Definiert in Zeile 178 der Datei laden.hh.

5.4.4.3 warenkorb

```
vector<waren> Kunde::warenkorb  [private]
```

Definiert in Zeile 179 der Datei laden.hh.

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.

Definiert in Zeile 355 der Datei lager.hh.

5.5.2 Dokumentation der benutzerdefinierten Datentypen

5.5.2.1 artikelMap

```
typedef map<string, Artikel *> Lager::artikelMap
```

Typdefinition fuer eine Map von Artikelnummern zu Artikeln.

Definiert in Zeile 373 der Datei lager.hh.

5.5.3 Beschreibung der Konstruktoren und Destruktoren

5.5.3.1 Lager()

```
Lager::Lager ( )  [default]
```

Standardkonstruktor fuer die Klasse Lager.

5.5.3.2 ~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.

Definiert in Zeile 26 der Datei lager.cc.

5.5.4 Dokumentation der Elementfunktionen

5.5.4.1 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.

Definiert in Zeile 86 der Datei lager.cc.

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

5.5.4.2 getMap()

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

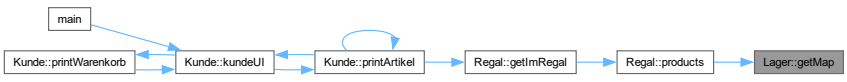
Gibt die gesamte Map von Artikelnummern zu Artikeln zurueck.

Rückgabe

Die Map von Artikelnummern zu Artikeln.

Definiert in Zeile 91 der Datei lager.cc.

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



5.5.4.3 readFile()

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

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

Parameter

filename	Der Dateiname der Eingabedatei.

Definiert in Zeile 34 der Datei lager.cc.

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

5.5.4.4 updateArtikel()

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

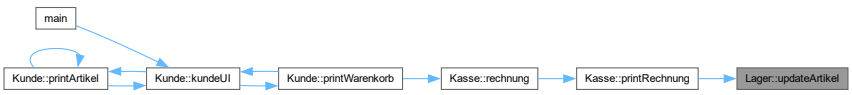
Aktualisiert die Informationen fuer einen Artikel in der Map.

Parameter

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

Definiert in Zeile 70 der Datei lager.cc.

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



5.5.4.5 write() [1/2]

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

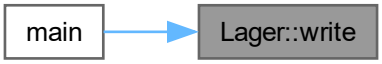
Schreibt die Artikelinformationen in den angegebenen Ausgabeostream.

Parameter

os	Der Ausgabeostream, in den die Informationen geschrieben werden.

Definiert in Zeile 62 der Datei lager.cc.

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



5.5.4.6 write() [2/2]

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

Schreibt die Artikelinformationen in eine Datei.

Parameter

filename	Der Dateiname der Ausgabedatei.

Definiert in Zeile 75 der Datei lager.cc.

5.5.5 Dokumentation der Datenelemente

5.5.5.1 lagerMap

```
artikelMap Lager::lagerMap  [private]
```

< Die Map von Artikelnummern zu Artikeln im Lager.

Definiert in Zeile 416 der Datei lager.hh.

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.
- template<class OutputIterator >
  void products (OutputIterator out) const

  Template-Funktion zum Abrufen von Produkten im Regal.
- 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.

Definiert in Zeile 30 der Datei laden.hh.

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.

Definiert in Zeile 27 der Datei laden.cc.

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.

Definiert in Zeile 31 der Datei laden.cc.

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

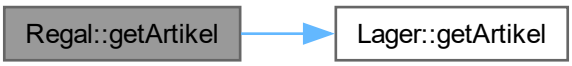
num	Die Artikelnummer des gesuchten Artikels.

Rückgabe

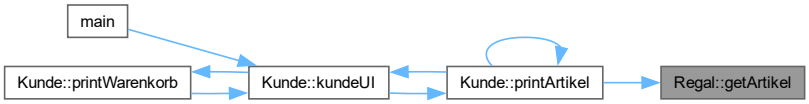
Der gefundene Artikel im Regal.

Definiert in Zeile 39 der Datei laden.cc.

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.

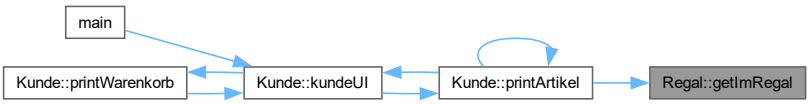
Rückgabe

Ein Vektor von Artikelnummern im Regal.

Definiert in Zeile 57 der Datei laden.cc.

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.

Definiert in Zeile 63 der Datei laden.cc.

5.6.3.4 getName()

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

Gibt den Namen des Regals zurueck.

Rückgabe

Der Name des Regals.

Definiert in Zeile 36 der Datei laden.cc.

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.

Definiert in Zeile 37 der Datei laden.cc.

5.6.3.6 products()

```
template<class OutputIterator >
void Regal::products (
            OutputIterator out ) const
```

Template-Funktion zum Abrufen von Produkten im Regal.

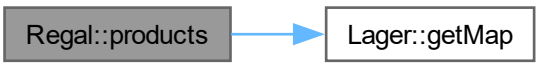
Diese Funktion ruft Produkte aus dem Regal ab und gibt die Ergebnisse an den angegebenen Ausgabeiterator aus.

Parameter

out	Der Ausgabeiterator, der das Ziel faer die Produkte repraesentiert.

Definiert in Zeile 42 der Datei laden.cc.

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



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

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.

Definiert in Zeile 65 der Datei laden.cc.

5.6.5 Dokumentation der Datenelemente

5.6.5.1 lager

```
Lager& Regal::lager  [private]
```

Die Menge der Warengruppen, die dem Regal zugeordnet sind.

Definiert in Zeile 112 der Datei laden.hh.

5.6.5.2 regalname

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

< Der Name des Regals.

Referenz auf das Lager, zu dem das Regal gehoert.

Definiert in Zeile 110 der Datei laden.hh.

5.6.5.3 waren

```
std::set<int> Regal::waren  [private]
```

Definiert in Zeile 114 der Datei laden.hh.

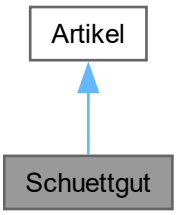
Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

- /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

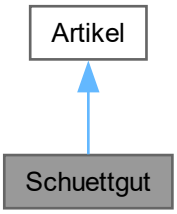
5.7 Schuettgut Klassenreferenz

`#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.

Definiert in Zeile 249 der Datei lager.hh.

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.

Definiert in Zeile 221 der Datei lager.cc.

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).

Definiert in Zeile 225 der Datei lager.cc.

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.

Definiert in Zeile 229 der Datei lager.cc.

5.7.3.2 setLosgroesse()

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

Setzt die Losgroesse des Schuettgut-Artikels.

Parameter

groesse	Die neue Losgroesse.

Definiert in Zeile 230 der Datei lager.cc.

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

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.

Definiert in Zeile 234 der Datei lager.cc.

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

5.7.4 Dokumentation der Datenelemente

5.7.4.1 losgroesse

```
double Schuettgut::losgroesse  [private]
```

Definiert in Zeile 251 der Datei lager.hh.

Die Dokumentation für diese Klasse wurde erzeugt aufgrund der Dateien:

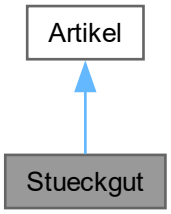
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.hh
- /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/Aufgabe_3/src/lager.cc

5.8 Stueckgut Klassenreferenz

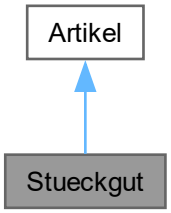
Die Klasse "Stueckgut" erbt von der Klasse "Artikel" und spezialisiert sie fuer Stueckgut-Artikel.

```
#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.

Definiert in Zeile 228 der Datei lager.hh.

5.8.2 Beschreibung der Konstruktoren und Destruktoren

5.8.2.1 Stueckgut() [1/2]

```
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).

Definiert in Zeile 215 der Datei lager.cc.

5.8.2.2 Stueckgut() [2/2]

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

Definiert in Zeile 218 der Datei lager.cc.

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.

Definiert in Zeile 164 der Datei laden.hh.

5.9.2 Dokumentation der Datenelemente

5.9.2.1 artikelnummer

```
string Kunde::waren::artikelnummer
```

Die Artikelnummer.

Definiert in Zeile 165 der Datei laden.hh.

5.9.2.2 menge

```
double Kunde::waren::menge
```

Die Menge des Artikels im Warenkorb.

Definiert in Zeile 166 der Datei laden.hh.

Die Dokumentation für diese Struktur wurde erzeugt aufgrund der Datei:

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

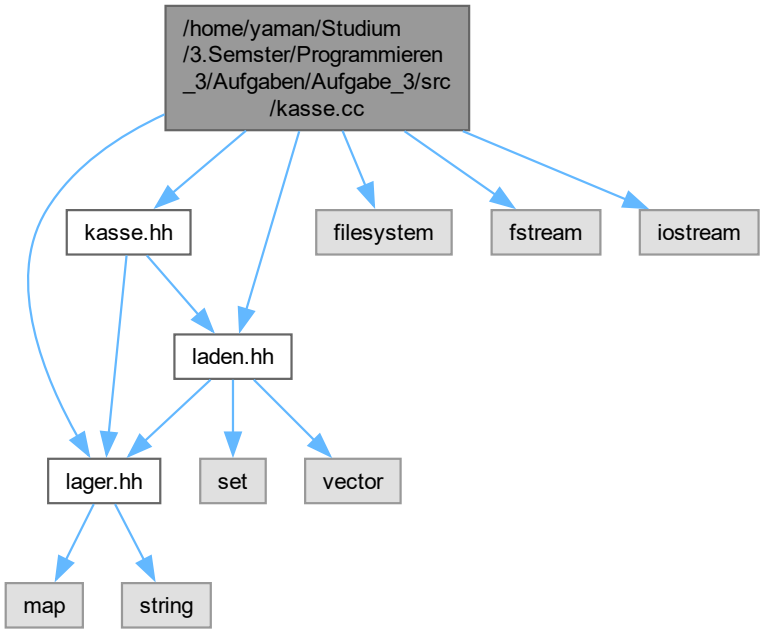
Kapitel 6

Datei-Dokumentation

6.1 /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/↩ Aufgabe_3/src/kasse.cc-Dateireferenz

```
#include "kasse.hh"
#include "laden.hh"
#include "lager.hh"
#include <filesystem>
#include <fstream>
#include <iostream>
```

Include-Abhängigkeitsdiagramm für kasse.cc:



Makrodefinitionen

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

6.1.1 Makro-Dokumentation

6.1.1.1 CLEAR

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

Definiert in Zeile 31 der Datei kasse.cc.

6.2 kasse.cc

gehe zur Dokumentation dieser Datei

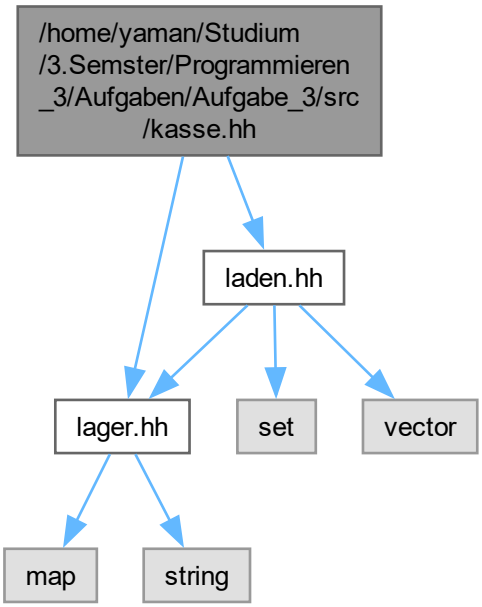
```
00001
00015 #include "kasse.hh"
00016 #include "laden.hh"
00017 #include "lager.hh"
00018 // #include <ctime>
00019 #include <filesystem>
00020 #include <fstream>
00021 // #include <iomanip>
00022 // #include <ios>
00023 #include <iostream>
00024 // #include <ostream>
00025 // #include <string>
00026 // #include <unistd.h>
00027 // #include <vector>
00028
00029 using namespace std;
00030
00031 #define CLEAR u8"\033[2J\033[1;1H"
00032 Kasse::Kasse(Kunde const &kunde, Lager &lager) : kunde(kunde), lager(lager) {}
00033
00034 void Kasse::rechnung(ostream &os) {
00035
00036   time_t now = time(0);
00037   struct tm currentTime;
00038   localtime_r(&now, &currentTime);
00039   int year = currentTime.tm_year + 1900; // Jahr seit 1900
00040   int month = currentTime.tm_mon + 1;    // Monat von 0 bis 11
00041   int day = currentTime.tm_mday;         // Tag des Monats
00042
00043   string date = to_string(year) + "-" + to_string(month) + "-" + to_string(day);
00044   string rechnungsnummer = to_string(month) + to_string(year) + to_string(day);
00045
00046   filesystem::path currentDir = filesystem::current_path();
00047   string dateiname = "rechnungen/" + date + "_" + kunde.getName() + ".txt";
00048
00049   if (!filesystem::exists("rechnungen")) {
00050     filesystem::create_directory("rechnungen");
00051   }
00052   ofstream datei(dateiname);
00053
00054   cout « CLEAR;
00055   printRechnung(os, date, rechnungsnummer, true);
00056   string wahl;
00057   while (true) {
00058     cin » wahl;
00059     if (wahl[0] == 'q') {
00060       os « "\n" « endl;
00061       break;
00062     }
00063     if (wahl[0] == 'p') {
00064       os « "\n" « endl;
00065       // string dateiname = kunde.getName() + ".txt";
00066       ofstream datei(dateiname);
00067       if (datei.is_open()) {
00068         printRechnung(datei, date, rechnungsnummer, false);
00069         datei.close();
00070       }
00071       cout « CLEAR;
```

```
00072       cout « "Rechnung liegt hier: " « currentDir « "/" « dateiname « endl;
00073       break;
00074     }
00075     cout « "Falsche Eingabe!" « endl;
00076   }
00077 }
00078
00079 void Kasse::printRechnung(ostream &os, const string &date,
00080                           const string &rechnungsnummer, bool print_auswahl) {
00081   string one_long = "-----------------------------------------------------";
00082   string double_short = "-----------------------\n-----------------------";
00083   double sum = 0;
00084
00085   os « "Rechnung des Warenwirtschaftssystems" « endl;
00086   os « "Rechnungsnummer: " « rechnungsnummer « endl;
00087   os « "Kunde: " « kunde.getName() « endl;
00088   os « "Rechnungdatum: " « date « "\n" « endl;
00089
00090   for (Kunde::waren ware : kunde.getWarenkorb()) {
00091     Artikel artikel = lager.getArtikel(ware.artikelnummer);
00092     os « one_long « endl;
00093     os « artikel.getName() « "\t" « artikel.getNormpreis() « " x ";
00094     os « ware.menge « " " « artikel.getStrMasseinheit() « "/EUR";
00095     os « "\t" « artikel.getNormpreis() * ware.menge « "EUR" « endl;
00096     artikel.setLagerbestand(artikel.getLagerbestand() - ware.menge);
00097     if (print_auswahl == false) {
00098       lager.updateArtikel(ware.artikelnummer, new Artikel(artikel));
00099     }
00100     sum += artikel.getNormpreis() * ware.menge;
00101   }
00102
00103   os « "\n"
00104      « "Summe Netto:\t" « sum « "EUR" « endl;
00105   os « "MwSt. 19%:\t" « sum * 0.19 « "EUR" « endl;
00106   os « "Gesamt:\t\t" « sum * (1 - 0.19) « "EUR" « endl;
00107   os « double_short « endl;
00108
00109   if (print_auswahl == true) {
00110     os « "Beenden q:" « endl;
00111     os « "Drucken p:" « endl;
00112     os « "Auswahl: ";
00113   }
00114 }
```

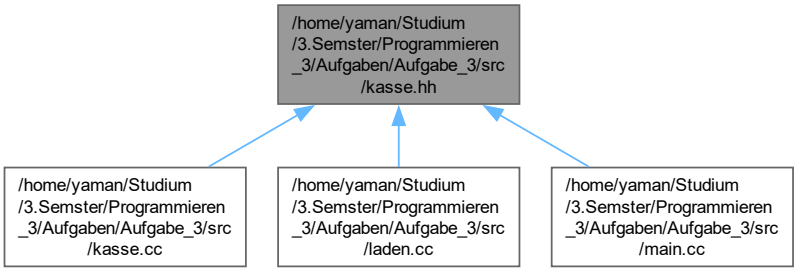
6.3 /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/↩ Aufgabe_3/src/kasse.hh-Dateireferenz

```
#include "laden.hh"
#include "lager.hh"
```

Include-Abhängigkeitsdiagramm für kasse.hh:



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



Klassen

- class Kasse

  Repraesentiert eine Kasse fuer Einkaeufe und erstellt Rechnungen.

6.4 kasse.hh

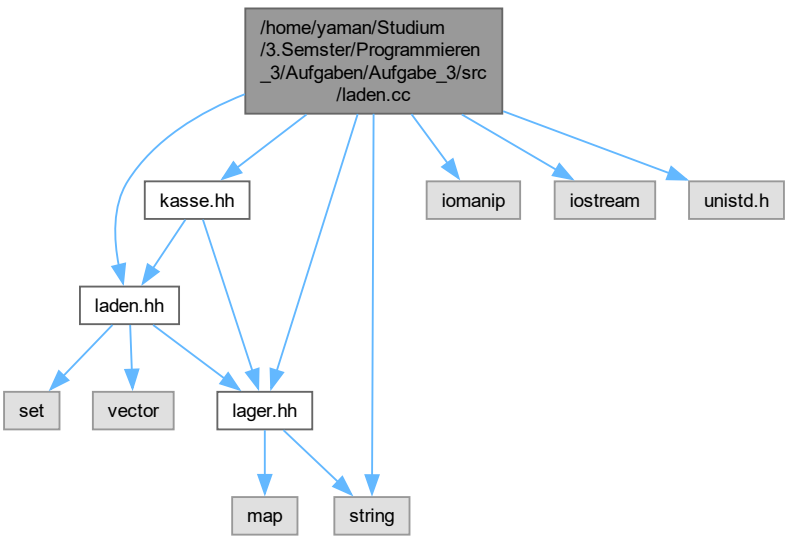
gehe zur Dokumentation dieser Datei

```
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
```

6.5 /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/↩ Aufgabe_3/src/laden.cc-Dateireferenz

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

Include-Abhängigkeitsdiagramm für laden.cc:



Makrodefinitionen

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

Funktionen

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

6.5.1 Makro-Dokumentation

6.5.1.1 CLEAR

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

Definiert in Zeile 25 der Datei laden.cc.

6.5.2 Dokumentation der Funktionen

6.5.2.1 operator<<()

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

Parameter

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

Rückgabe

Der Ausgabestrom.

Definiert in Zeile 65 der Datei laden.cc.

6.6 laden.cc

gehe zur Dokumentation dieser Datei

```
00001
00015 #include "laden.hh"
00016 #include "kasse.hh"
00017 #include "lager.hh"
00018 #include <iomanip>
00019 #include <iostream>
00020 #include <string>
00021 #include <unistd.h>
00022 // #include <vector>
00023
00024 using namespace std;
00025 #define CLEAR u8"\033[2J\033[1;1H"
00026
00027 Regal::Regal(string name, Lager &lager, int warengruppe)
00028     : regalname(name), lager(lager) {
00029   waren.insert(warengruppe);
00030 }
00031 Regal::Regal(string name, Lager &lager, std::set<int> warengruppen)
00032     : regalname(name), lager(lager) {
00033   waren.merge(warengruppen);
00034 }
```

```
00035
00036 string Regal::getName() const { return regalname; }
00037 std::set<int> Regal::getWaren() const { return waren; }
00038
00039 Artikel Regal::getArtikel(string num) const { return lager.getArtikel(num); }
00040
00041 template <class OutputIterator>
00042 void Regal::products(OutputIterator out) const {
00043   Lager::artikelMap map = lager.getMap();
00044   for (int ware : waren) {
00045     Lager::artikelMap::iterator it = map.begin();
00046     while (it != map.end()) {
00047       int num = (*it->second).getGruppe();
00048       num /= 100;
00049       if (ware == num) {
00050         *out = it->first;
00051         out++;
00052       }
00053       it++;
00054     }
00055   }
00056 }
00057 vector<string> Regal::getImRegal() const {
00058   vector<std::string> imRegal;
00059   products(std::back_insert_iterator(imRegal));
00060   return imRegal;
00061 }
00062
00063 Lager &Regal::getLager() { return lager; }
00064
00065 ostream &operator«(ostream &os, Regal regal) {
00066   vector<string> imRegal = regal.getImRegal();
00067   int i = 0;
00068   for (auto num : imRegal) {
00069     // cout « Lager(regal.lager).getArtikel(num) « endl;
00070     Artikel artikel = regal.lager.getArtikel(num);
00071     i++;
00072     cout « setw(5) « "";
00073     cout « i « setw(9) « ":" « left;
00074     cout « setw(30) « artikel.getName();
00075     cout « setw(20) « artikel.getLagerbestand();
00076     cout « artikel.getVerkaufpreis() « "/"
00077          « artikel.getVerkaufpreis() / artikel.getNormpreis() « setw(20)
00078          « artikel.getStrMasseinheit() « endl;
00079   }
00080   return os;
00081 }
00082
00083 Kunde::Kunde(string name, vector<Regal> const &regale)
00084     : name(name), regale(regale) {}
00085
00086 vector<Kunde::waren> Kunde::getWarenkorb() const { return warenkorb; }
00087
00088 string Kunde::getName() const { return name; }
00089
00090 void Kunde::kundeUI() {
00091   string wahl;
00092   size_t wahlNum;
00093   cout « CLEAR;
00094   int i = 0;
00095   cout « "Warenkorb: " « warenkorb.size() « endl;
00096   cout « "Waehlen Sie einen Regal aus\n" « left « endl;
00097   cout « setw(2) « "";
00098   cout « "Wahl" « setw(9) « ":" « left;
00099   cout « setw(30) « "Bezeichnung" « endl;
00100   cout « setw(5) « "";
00101   cout « "0" « setw(9) « ":" « left;
00102
00103   cout « "Warenkorb" « left;
00104   cout « endl;
00105   for (auto regal : regale) {
00106     i++;
00107     cout « setw(5) « "";
00108     cout « i « setw(9) « ":" « left;
00109     cout « regal.getName() « endl;
00110   }
00111   cout « setw(5) « "";
00112   cout « "q" « setw(9) « ":" « left;
00113
00114   cout « "Beenden" « left;
00115   cout « endl;
00116   cout « endl;
00117   while (true) {
00118     cout « "Auswahl: ";
00119     cin » wahl;
00120     if (wahl[0] == 'q')
00121       return;
```

```
00122     try {
00123       wahlNum = stoi(wahl);
00124       if (wahlNum > regale.size()) {
00125         cout « "Falsche Eingabe!" « endl;
00126       } else if (wahlNum == 0) {
00127         printWarenkorb();
00128         break;
00129       } else {
00130         printArtikel(wahlNum - 1);
00131         break;
00132       }
00133     } catch (const std::exception &) {
00134       cout « "Falsche Eingabe!!" « endl;
00135     }
00136   }
00137 }
00138
00139 void Kunde::printArtikel(int num) {
00140   cout « CLEAR;
00141   string wahl1, wahl2;
00142   size_t wahl1Num;
00143   double wahl2num;
00144   cout « "Warenkorb: " « warenkorb.size() « endl;
00145   cout « "Waehlen Sie einen Artikel aus\n" « left « endl;
00146   cout « setw(15) « "";
00147   cout « setw(30) « "Bezeichnung";
00148   cout « setw(20) « "Lagerbestand";
00149   cout « setw(20) « "Preis/Einheit" « endl;
00150
00151   cout « regale[num];
00152   cout « setw(5) « "";
00153   cout « "." « setw(9) « ":" « left;
00154   cout « "Zurueck" « left;
00155   cout « endl;
00156   cout « setw(5) « "";
00157   cout « "q" « setw(9) « ":" « left;
00158   cout « "Beenden" « left;
00159   cout « endl;
00160   cout « endl;
00161   while (true) {
00162     cout « "Auswahl: ";
00163     cin » wahl1;
00164     if (wahl1[0] == 'q') {
00165       break;
00166     }
00167     if (wahl1[0] == '.') {
00168       kundeUI();
00169       break;
00170     }
00171     try {
00172
00173       wahl1Num = stoi(wahl1);
00174       if (wahl1Num > Regal(regale[num]).getImRegal().size()) {
00175       } else {
00176         wahl1Num--;
00177         string artnum = Regal(regale[num]).getImRegal()[wahl1Num];
00178         Artikel artikel = Regal(regale[num]).getArtikel(artnum);
00179         cout « artikel.getName() « endl;
00180         cout « "Geben Sie die Menge" « endl;
00181         cin » wahl2;
00182         wahl2num = stof(wahl2);
00183         if (wahl2num <= artikel.getLagerbestand()) {
00184           warenkorb.push_back({artnum, wahl2num});
00185           cout « CLEAR;
00186           float menge = warenkorb[warenkorb.size() - 1].menge;
00187           cout « menge « " * " « artikel.getName() « endl;
00188           sleep(1);
00189           printArtikel(num);
00190           break;
00191         }
00192       }
00193     } catch (const std::exception &) {
00194     }
00195     cout « "Falsche Eingabe!" « endl;
00196   }
00197 }
00198 void Kunde::printWarenkorb() {
00199   string wahl;
00200   int i = 0;
00201   cout « CLEAR;
00202   cout « "Warenkorb: " « warenkorb.size() « endl;
00203   cout « "Waehlen Sie aus\n" « left « endl;
00204   for (auto ware : warenkorb) {
00205     Artikel artikel = regale[0].getArtikel(ware.artikelnummer);
00206     i++;
00207     cout.imbue(locale("de_DE.UTF-8"));
00208     cout « setw(5) « "";
```

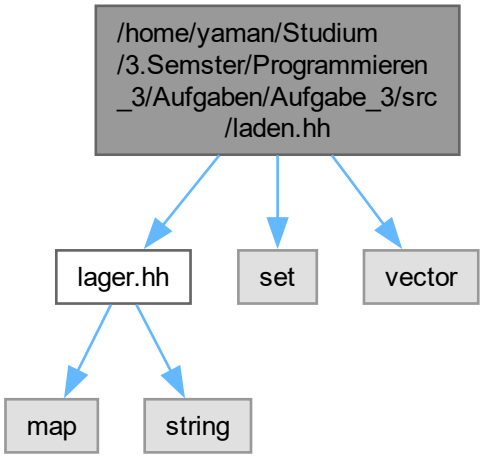
```
00209     cout « i « setw(9) « ":" « left;
00210     cout « setw(30) « artikel.getName();
00211     cout « setw(6) « artikel.getVerkaufpreis() « "/" « setw(4)
00212          « artikel.getVerkaufpreis() / artikel.getNormpreis();
00213     cout « setw(20) « artikel.getStrMasseinheit();
00214     cout « setw(20) « ware.menge;
00215     cout « setw(20) « showbase « (artikel.getNormpreis() * ware.menge)
00216          « endl;
00217   }
00218   cout « setw(5) « "";
00219   cout « "k" « setw(9) « ":" « left;
00220   cout « "Kasse" « left;
00221   cout « endl;
00222   cout « setw(5) « "";
00223   cout « "." « setw(9) « ":" « left;
00224   cout « "Zurueck" « left;
00225   cout « endl;
00226   cout « setw(5) « "";
00227   cout « "q" « setw(9) « ":" « left;
00228   cout « "Beenden" « left « endl;
00229   cout « "Auswahl: ";
00230   while (true) {
00231     cin » wahl;
00232     if (wahl[0] == 'q') {
00233       break;
00234     }
00235     if (wahl[0] == '.') {
00236       kundeUI();
00237       break;
00238     }
00239     if (wahl[0] == 'k') {
00240       Kasse kasse(*this, Regal(regale[0]).getLager());
00241       kasse.rechnung(cout);
00242       break;
00243     }
00244     cout « "Falsche Eingabe!" « endl;
00245   }
00246 }
```

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

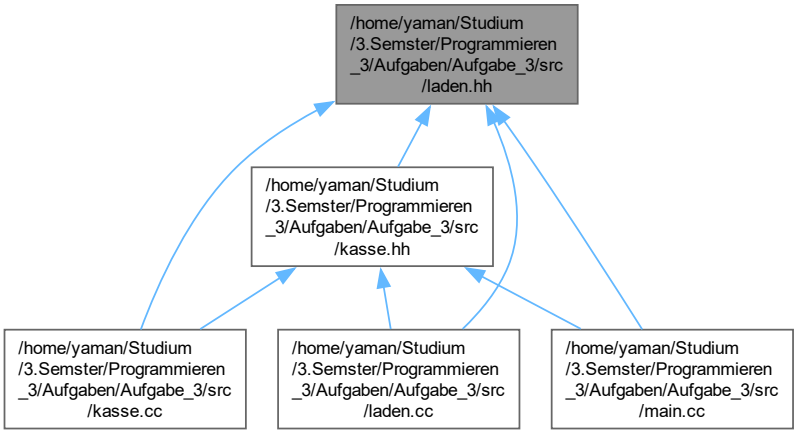
Enthaelt die Deklaration der Klasse Kasse.

```
#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.

6.7.1 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

Definiert in Datei laden.hh.

6.8 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
00059   template <class OutputIterator>
00060   void products(OutputIterator out) const;
00061
00067   Lager &getLager();
00068
00074   string getName() const;
00075
00082   set<int> getWaren() const;
00083
00090   Artikel getArtikel(string num) const;
00091
00097   vector<string> getImRegal() const;
00098
00106   friend ostream &operator«(ostream &os, Regal regal);
00107
00108 private:
00110   string regalname;
00112   Lager &lager;
00114   std::set<int> waren;
00115 };
00116
00125 class Kunde {
00126 public:
00135   Kunde(string name, vector<Regal> const &regale);
00136
00140   void kundeUI();
00141
00147   string getName() const;
00148
00154   void printArtikel(int num);
00155
00159   void printWarenkorb();
00160
00164   typedef struct {
00165     string artikelnummer;
00166     double menge;
00167   } waren;
00168
00174   vector<waren> getWarenkorb() const;
00175
00176 private:
00177   string name;
00178   vector<Regal> const &regale;
00179   vector<waren> warenkorb;
00180 };
00181
00182 #endif // !LADEN_HH
```

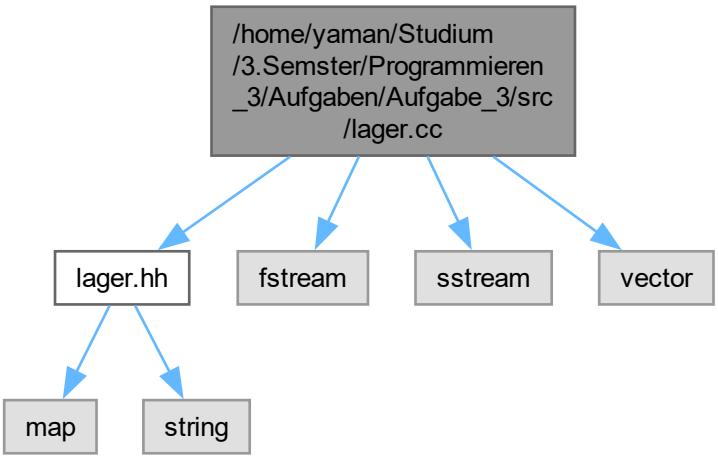
6.9 /home/yaman/Studium/3.Semster/Programmieren_3/Aufgaben/↩ Aufgabe_3/src/lager.cc-Dateireferenz

Implementierung der Lagerverwaltungsfunktionen.

```
#include "lager.hh"
#include <fstream>
#include <sstream>
```

```
#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.

6.9.1 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

Definiert in Datei lager.cc.

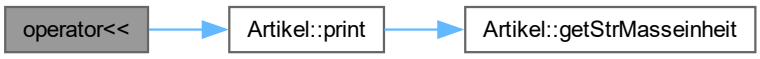
6.9.2 Dokumentation der Funktionen

6.9.2.1 operator<<()

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

Definiert in Zeile 138 der Datei lager.cc.

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



6.9.2.2 operator>>()

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

ueberladen des Eingabeoperators fuer die Artikelklasse.

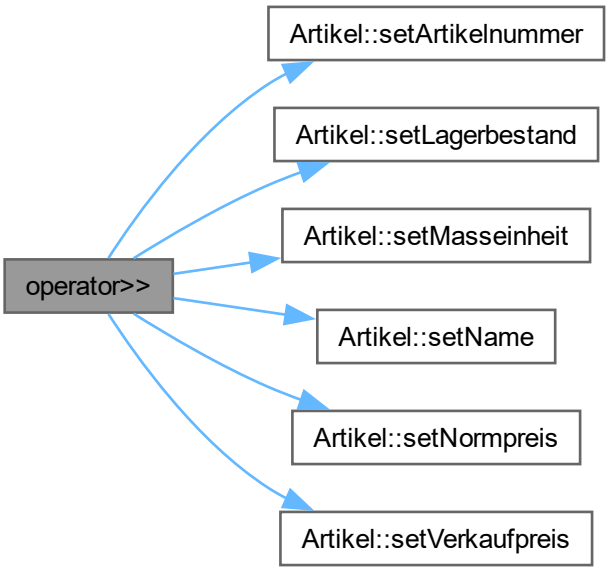
Diese Funktion ermoeglicht das Einlesen von Artikelinformationen mit dem Eingabeoperator '>>'.

Parameter

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

Definiert in Zeile 142 der Datei lager.cc.

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

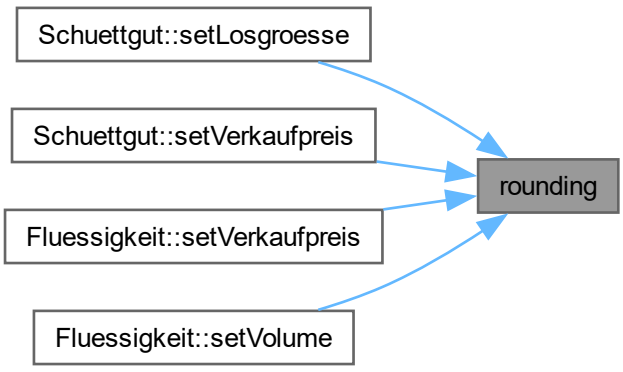


6.9.2.3 rounding()

```
static double rounding (
             double num )  [static]
```

Definiert in Zeile 256 der Datei lager.cc.

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



6.10 lager.cc

gehe zur Dokumentation dieser Datei

```
00001
00015 #include "lager.hh"
00016 // #include <cmath>
00017 #include <fstream>
00018 // #include <iostream>
00019 // #include <map>
00020 #include <sstream>
00021 // #include <string>
00022 #include <vector>
00023
00024 static double rounding(double);
00025
00026 Lager::~Lager() {
00027   artikelMap::iterator it = lagerMap.begin();
00028   while (it != lagerMap.end()) {
00029     delete (it->second);
00030     it++;
00031   }
00032 }
00033
00034 void Lager::readFile(string filename) {
00035   ifstream file(filename);
00036   if (file.is_open()) {
00037     Artikel tmp;
00038     do {
00039       try {
00040         file » tmp;
00041         switch (tmp.getMasseinheit()) {
00042         case 0:
00043           lagerMap.insert({tmp.getArtikelnummer(), new Stueckgut(tmp)});
00044           break;
00045         case 1:
00046           lagerMap.insert({tmp.getArtikelnummer(), new Schuettgut(tmp)});
00047           break;
00048         case 2:
00049           lagerMap.insert({tmp.getArtikelnummer(), new Fluessigkeit(tmp)});
00050           break;
00051         }
00052       } catch (const int &ex) {
00053       } catch (std::invalid_argument const &ex) {
00054       }
00055     } while (!file.eof());
00056     file.close();
00057   } else {
00058     exit(EXIT_FAILURE);
00059   }
00060 }
00061
00062 void Lager::write(ostream &os) {
00063   artikelMap::iterator it = lagerMap.begin();
00064   while (it != lagerMap.end()) {
00065     os « *it->second « endl;
00066     it++;
00067   }
00068 }
00069
00070 void Lager::updateArtikel(string num, Artikel *artikel) {
00071   delete (lagerMap[num]);
00072   lagerMap[num] = artikel;
00073 }
00074
00075 void Lager::write(string filename) {
00076   ofstream file(filename);
00077   if (file.is_open()) {
00078     artikelMap::iterator it = lagerMap.begin();
00079     while (it != lagerMap.end()) {
00080       file « *it->second « endl;
00081       it++;
00082     }
00083   }
00084 }
00085
00086 Artikel Lager::getArtikel(string artikelnummer) const {
00087
00088   return *artikelMap(lagerMap)[artikelnummer];
00089 }
00090
00091 Lager::artikelMap Lager::getMap() { return lagerMap; }
00092
00093 Artikel::Artikel() {}
00094 Artikel::Artikel(string name, string num, unsigned int bestand,
00095                  masseinheit einheit, preis vp, preis np)
```

```
00096     : artikelname(name), artikelnummer(num), lagerbestand(bestand),
00097       einheit(einheit), verkaufpreis(vp), normpreis(np) {}
00098 Artikel::~Artikel() {}
00099
00100 // void Artikel::setGruppe(Warengruppen g) { gruppe = g; }
00101 string Artikel::getName() const { return artikelname; }
00102 string Artikel::getArtikelnummer() const { return artikelnummer; }
00103 unsigned int Artikel::getLagerbestand() const { return lagerbestand; }
00104 masseinheit Artikel::getMasseinheit() const { return einheit; }
00105 string Artikel::getStrMasseinheit() const {
00106   switch (einheit) {
00107   case 0:
00108     return "Stk";
00109   case 1:
00110     return "kg";
00111   case 2:
00112     return "l";
00113   default:
00114     return "None";
00115   }
00116 }
00117
00118 preis Artikel::getVerkaufpreis() const { return verkaufpreis; }
00119 preis Artikel::getNormpreis() const { return normpreis; }
00120 int Artikel::getGruppe() const {
00121   string artnum = artikelnummer;
00122   artnum = artnum.erase(4);
00123   return stoi(artnum);
00124 }
00125
00126 void Artikel::setName(string name) { artikelname = name; }
00127 void Artikel::setArtikelnummer(string num) { artikelnummer = num; }
00128 void Artikel::setLagerbestand(unsigned int bestand) { lagerbestand = bestand; }
00129 void Artikel::setMasseinheit(masseinheit einheit) { this->einheit = einheit; }
00130 void Artikel::setVerkaufpreis(preis vp) { verkaufpreis = vp; }
00131 void Artikel::setNormpreis(preis np) { normpreis = np; }
00132
00133 std::ostream &Artikel::print(std::ostream &os) {
00134   return os « artikelname « "|" « artikelnummer « "|" « lagerbestand « "|"
00135            « verkaufpreis « "|" « getStrMasseinheit() « "|" « normpreis;
00136 }
00137
00138 std::ostream &operator«(std::ostream &os, Artikel produkt) {
00139   return produkt.print(os);
00140 }
00141
00142 void operator»(istream &is, Artikel &produkt) {
00143   vector<string> beschreibung;
00144   string text, name, num;
00145   int bestand = 0;
00146   preis vp = 0, np = 0;
00147   masseinheit einheit;
00148
00149   getline(is, text);
00150   stringstream ss(text);
00151
00152   if (!text[0]) {
00153     throw(-1);
00154   }
00155   text = "";
00156   for (size_t i = 0; getline(ss, text, '|') && i < 6; i++) {
00157     beschreibung.push_back(text);
00158   }
00159   if (beschreibung.size() < 5)
00160     throw -1;
00161   name = beschreibung[0];
00162   num = beschreibung[1];
00163
00164   if (beschreibung[4] == "kg")
00165     einheit = kg;
00166   else if (beschreibung[4] == "l")
00167     einheit = l;
00168   else if (beschreibung[4] == "stk")
00169     einheit = stk;
00170   else {
00171     einheit = stk;
00172   }
00173
00174   for (size_t i = 1; i < 6; i++) {
00175     for (size_t j = 0; j < beschreibung[i].length(); j++) {
00176       if (beschreibung[i][j] == ' ') {
00177         beschreibung[i].erase(beschreibung[i].begin() + j);
00178         j--;
00179       }
00180     }
00181   }
00182
```

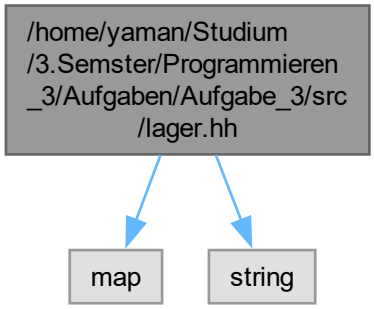
```
00183   if (name == "" || num == "" || num.length() != 10) {
00184     throw(-1);
00185   }
00186   if (beschreibung[3] == "" && beschreibung[4] == "") {
00187     throw(-1);
00188   }
00189
00190   if (beschreibung[2] != "") {
00191     bestand = stoi(beschreibung[2]);
00192   } else {
00193     bestand = 0;
00194   }
00195   if (beschreibung[3] != "") {
00196     vp = stof(beschreibung[3]);
00197   }
00198
00199   if (beschreibung.size() > 5 && beschreibung[5] != "") {
00200     np = stof(beschreibung[5]);
00201   }
00202   if (vp == 0)
00203     vp = np;
00204   if (np == 0)
00205     np = vp;
00206
00207   produkt.setMasseinheit(einheit);
00208   produkt.setName(beschreibung[0]);
00209   produkt.setArtikelnummer(beschreibung[1]);
00210   produkt.setLagerbestand(bestand);
00211   produkt.setVerkaufpreis(vp);
00212   produkt.setNormpreis(np);
00213 }
00214
00215 Stueckgut::Stueckgut(Artikel produkt)
00216     : Stueckgut(produkt.getName(), produkt.getArtikelnummer(),
00217                 produkt.getVerkaufpreis(), produkt.getLagerbestand()) {}
00218 Stueckgut::Stueckgut(string name, string num, preis vp, unsigned int bestand)
00219     : Artikel(name, num, bestand, stk, vp, vp) {}
00220
00221 Schuettgut::Schuettgut(Artikel produkt)
00222     : Schuettgut(produkt.getName(), produkt.getArtikelnummer(),
00223                  rounding(produkt.getVerkaufpreis() / produkt.getNormpreis()),
00224                  produkt.getNormpreis(), produkt.getLagerbestand()) {}
00225 Schuettgut::Schuettgut(string name, string num, double groesse, preis np,
00226                        unsigned int bestand)
00227     : Artikel(name, num, bestand, kg, (groesse * np), np), losgroesse(groesse) {
00228 }
00229 double Schuettgut::getLosgroesse() const { return losgroesse; }
00230 void Schuettgut::setLosgroesse(double groesse) {
00231   losgroesse = groesse;
00232   verkaufpreis = rounding(verkaufpreis);
00233 }
00234 void Schuettgut::setVerkaufpreis(preis vp) {
00235   verkaufpreis = vp;
00236   losgroesse = rounding(losgroesse);
00237 }
00238
00239 Fluessigkeit::Fluessigkeit(Artikel produkt)
00240     : Fluessigkeit(produkt.getName(), produkt.getArtikelnummer(),
00241                    rounding(produkt.getVerkaufpreis() / produkt.getNormpreis()),
00242                    produkt.getNormpreis(), produkt.getLagerbestand()) {}
00243 Fluessigkeit::Fluessigkeit(string name, string num, double vol, preis np,
00244                            unsigned int bestand)
00245     : Artikel(name, num, bestand, l, (vol * np), np), volume(vol) {}
00246 double Fluessigkeit::getVolume() const { return volume; }
00247 void Fluessigkeit::setVolume(double vol) {
00248   volume = vol;
00249   verkaufpreis = rounding(verkaufpreis);
00250 }
00251 void Fluessigkeit::setVerkaufpreis(preis vp) {
00252   verkaufpreis = vp;
00253   volume = rounding(volume);
00254 }
00255
00256 static double rounding(double num) {
00257   num *= 100;
00258   num += 0.5;
00259   num = int(num);
00260   num /= 100;
00261   return num;
00262 }
```

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

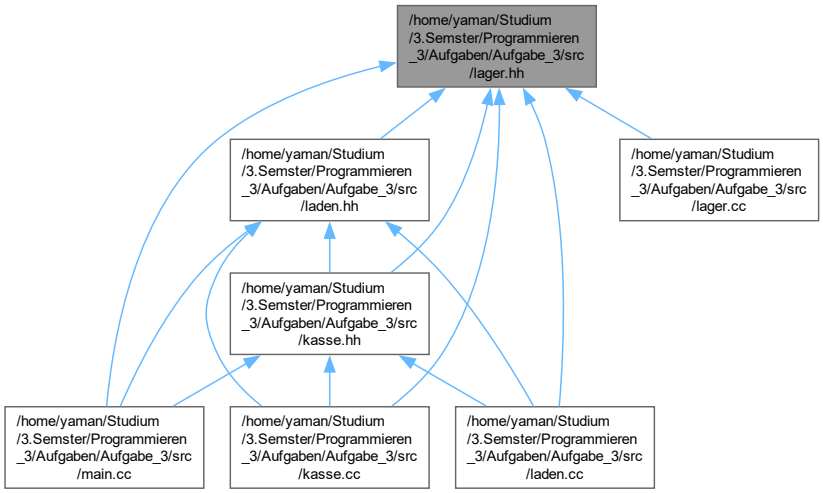
Definitionen der Lagerverwaltungsfunktionen.

```
#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.

6.11.1 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

Definiert in Datei lager.hh.

6.11.2 Dokumentation der benutzerdefinierten Typen

6.11.2.1 Nummer

```
typedef int Nummer
```

Definiert in Zeile 25 der Datei lager.hh.

6.11.2.2 preis

```
typedef double preis
```

Definiert in Zeile 24 der Datei lager.hh.

6.11.3 Dokumentation der Aufzählungstypen

6.11.3.1 masseinheit

```
enum masseinheit
```

Aufzählungswerte

stk	
kg	
l	

Definiert in Zeile 23 der Datei lager.hh.

6.11.4 Dokumentation der Funktionen

6.11.4.1 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.

6.11.4.2 operator>>()

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

ueberladen des Eingabeoperators fuer die Artikelklasse.

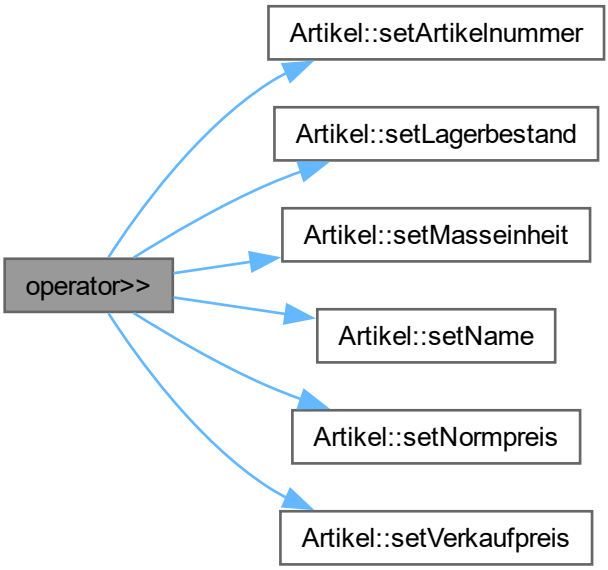
Diese Funktion ermoeglicht das Einlesen von Artikelinformationen mit dem Eingabeoperator '>>'.

Parameter

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

Definiert in Zeile 142 der Datei lager.cc.

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



6.12 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,
```

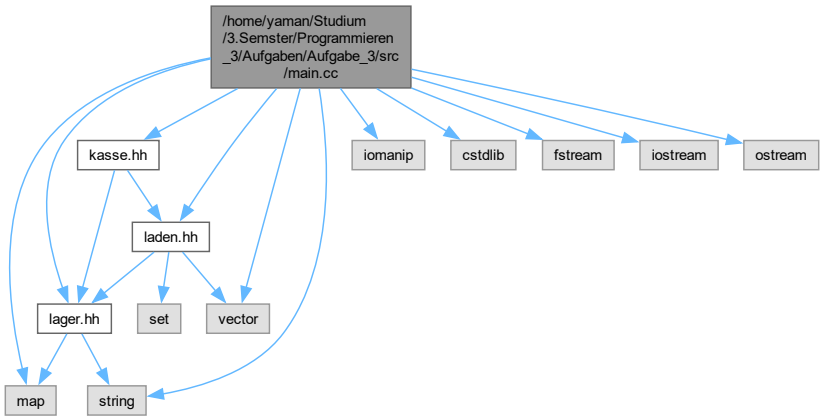
```
00272             unsigned int bestand = 1);
00273
00279   double getLosgroesse() const;
00280
00287   void setVerkaufpreis(preis vp);
00288
00294   void setLosgroesse(double groesse);
00295 };
00296
00304 class Fluessigkeit : public Artikel {
00305 private:
00306   double volume;
00307
00308 public:
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
```

6.13 /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 <iomanip>
#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.

6.13.1 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

Definiert in Datei main.cc.

6.13.2 Makro-Dokumentation

6.13.2.1 CLEAR

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

Definition fuer den Befehl zum Loeschen des Konsolenbildschirms.

Definiert in Zeile 34 der Datei main.cc.

6.13.3 Dokumentation der Funktionen

6.13.3.1 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

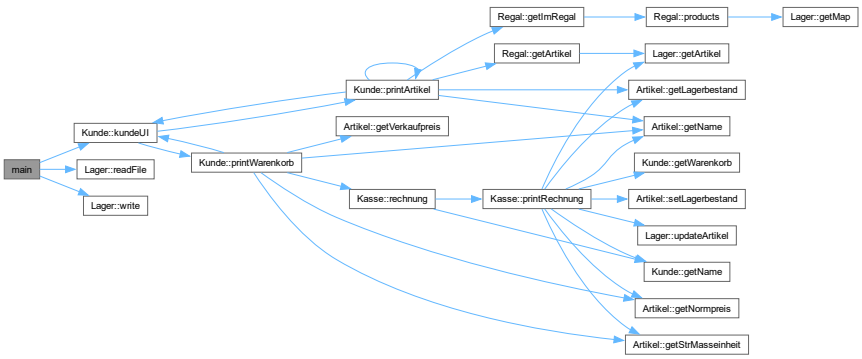
argc	Die Anzahl der Befehlszeilenargumente.
argv	Ein Array von Zeichenketten, das die Befehlszeilenargumente enthaelt.

Rückgabe

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

Definiert in Zeile 50 der Datei main.cc.

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



6.14 main.cc

gehe zur Dokumentation dieser Datei

```
00001
00016 #include "kasse.hh"
00017 #include "laden.hh"
00018 #include "lager.hh"
00019 #include <iomanip>
00020 #include <cstdlib>
00021 #include <fstream>
00022 #include <iostream>
00023 #include <map>
00024 #include <ostream>
00025 #include <string>
00026 #include <vector>
00027
00028 using namespace std;
00029
00034 #define CLEAR u8"\033[2J\033[1;1H"
00035
00050 int main(int argc, char *argv[]) {
00051   string filewrite = "";
00052   string fileread = "";
00053   char wahl;
00054   string vorname;
00055   string name;
00056   Lager lager;
00057   vector<Regal> regale;
00058   for (int i = 1; i < argc; i++) {
00059     string arg = argv[i];
00060     if (arg == "-o") {
00061       filewrite = argv[i + 1];
00062     } else if (string(argv[i]) == "-i") {
00063       fileread = argv[i + 1];
00064     }
00065   }
00066
00067   if (fileread == "") {
00068     exit(EXIT_FAILURE);
00069   }
00070
00071   // Lager aus der Eingabedatei lesen
00072   lager.readFile(fileread);
00073
00074   // Regale initialisieren
00075   Regal gemueseRegal(R"(Gemuese)", lager, {40, 41});
00076   Regal getraenkeRegal("Getraenke", lager, {43, 50, 55});
00077   Regal sonderRegal("Sonderartikel", lager, 10);
00078
00079   regale.push_back(gemueseRegal);
00080   regale.push_back(getraenkeRegal);
00081   regale.push_back(sonderRegal);
00082
00083   // Hauptbenutzerschleife
00084   while (true) {
00085     cout « CLEAR;
00086     cout « "Waehlen Sie aus!" « endl;
00087     cout « setw(20) « left « "\tEinkaufen: "
00088          « "n" « endl;
00089     cout « setw(20) « left « "\tFeierabend: "
00090          « "q" « endl;
00091     cout « "\nAuswahl:";
00092     cin » wahl;
00093
00094     // Programm beenden, wenn 'q' ausgewaehlt wird
00095     if (wahl == 'q') {
00096       break;
00097     }
00098
00099     // Einkaufen starten, wenn 'n' ausgewaehlt wird
00100     if (wahl == 'n') {
00101       cout « "Geben Ihre Name!" « endl;
00102       cin » vorname;
00103       cin » name;
00104       Kunde kunde(vorname + string(" ") + name, regale);
00105       kunde.kundeUI();
00106       cout « CLEAR;
00107     } else {
00108       break;
00109     }
00110   }
00111
00112   // Lager aktualisieren und speichern
00113   if (filewrite == "") {
00114     lager.write("out.txt");
```

```
00115   } else {
00116     lager.write(filewrite);
00117   }
00118   return 0;
00119 }
```

Index