Aufgabe Nr. 3 21.11.2023

Aufgabe 3.1

Falls noch nicht geschehen, modularisieren Sie Ihren Code so, dass sich die Deklarationen der öffentlichen Schnittstellen in Header-Dateien befinden und alles andere in den `.cc`/`.cpp`-Dateien versteckt wird. Hilfsfunktionen und globale Daten werden mit `static` versehen. Die Header-Dateien müssen Include-Guards haben.

Im Rahmen dieses Praktikums sollen außerdem zwei neue Module hinzukommen:

- Der Laden mit den Dateien `laden.cc` (bzw. `laden.cpp`) und `laden.hh` sowie
- die Kasse mit den Dateien `kasse.cc` (bzw. `kasse.cpp`) und `kasse.hh`.

Aufgabe 3.2

Nun sollen Produkte in Regalen präsentiert werden. Ein Regal wird als Klasse `Regal` im Modul Laden realisiert, der Konstruktor erhalte eine `const`-Referenz auf das Lager.

Vereinfachend gibt es in diesem Laden drei unterschiedliche Arten von Regalen:

1. Das Gemüseregal,
2. das Getränkeregal und
3. die Sonderartikel.

Um dies zu realisieren, erhält der Konstruktor zusätzlich eine Angabe der Warengruppe(n). Ein Grundgerüst für die Klasse `Regal` könnte also folgendermaßen aussehen:

```
class Regal {
public:
  Regal(const Lager &lager, Artikel::Nummer warengruppe);
  Regal(const Lager &lager, std::set<Artikel::Nummer> warengruppen);

  template <class OutputIterator>
  void products(OutputIterator out) const;
private:
  const Lager &lager;
  std::set<Artikel::Nummer> waren;
};
```

Implementieren Sie eine Methode

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

die für alle Produkte des jeweiligen Regals die Artikelnummer in den übergebenen Ausgabe-Iterator überträgt. Je nach Repräsentation der Artikelnummer in Ihrem Programmcode könnte ein Beispielaufruf folgendermaßen aussehen:

```
Regal gemueseRegal{lager, {40, 41}};

vector<Artikel::Nummer> imGemueseRegal;
gemueseRegal.products(std::back_insert_iterator(imGemueseRegal));
```

Beachten Sie, dass Templates für Klassen und Funktionen der öffentlichen Programmierschnittstelle eines Moduls in dessen Header-Datei implementiert sein müssen.

Aufgabe 3.3

Anhand der Artikel, die sich in den Regalen aus Aufgabe 3.2 befinden, soll im Hauptprogramm ein Dialog zur Produktauswahl erzeugt werden. Angezeigt werden soll:

1. Der Name des Regals (also zum Beispiel „Gemueseregal“; wo der Name des Regals verwaltet wird, entscheiden Sie selbst),
2. Name und verfügbare Menge der jeweiligen Artikel in dem Regal,
3. zusammen mit deren Verkaufspreis und dem Normpreis mit zugehöriger Einheit, wo sinnvoll.

Über einen einfachen Text-Dialog soll ein Kunde Artikel auswählen können. Als „Einkaufskorb“ diene die Klasse `Kunde`, die Artikelnummern und Menge aufnehmen soll. Ein Kunde kann nicht mehr aus den Regalen nehmen, als vorhanden ist.[1]

Aufgabe 3.4

An der Kasse erfolgt die Abrechnung der Waren, die ein Kunde aus dem Regal entnommen hat: Alle Waren aus dem Einkaufskorb des Kunden werden aus dem Lager entnommen und mit ihrem Verkaufspreis in Rechnung gestellt. Steht von einem Artikel weniger im Lager zur Verfügung als der Kunde „eingepackt“ hat, so erfolgt eine entsprechende Meldung, und es wird nur die zur Verfügung stehende Menge verbucht. Dem Kunden wird schließlich eine detaillierte Rechnung inklusive der Gesamtsumme präsentiert.

[1] Der Lagerbestand wird durch das Entnehmen aber **nicht** verändert — dies geschieht erst beim Ausbuchen an der Kasse. Damit berücksichtigt diese Modellierung nicht den Fall, dass andere Kunden auf das Regal zugreifen können, bevor die Ware korrekt ausgebucht worden ist. Anders als in der realen Welt würden die betreffenden Kunden also erst an der Kasse erfahren, dass das betreffende Produkt gar nicht im Regal stand.