Technische Dokumentation des Online-Shops

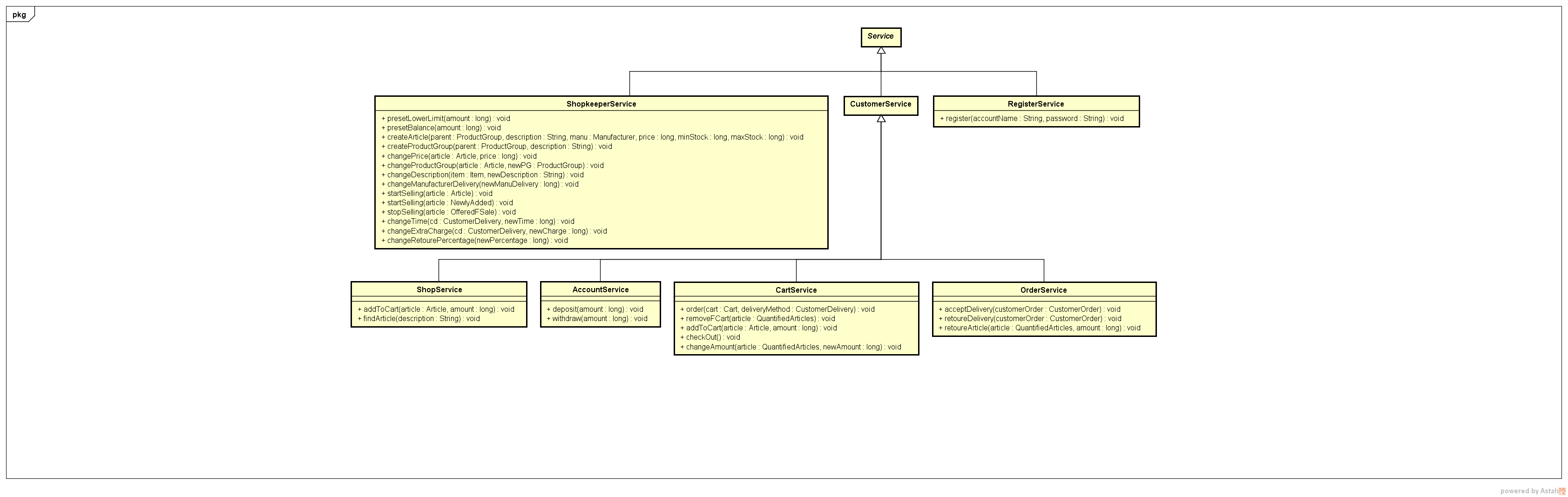
# Vorwort

Der Online-Shop besteht aus den beiden wesentlichen Sichten des Ladenbesitzers und des Kunden. Es gibt dabei genau einen Ladenbesitzer und beliebig viele Kunden, die jeweils über ein eigenes Kundenkonto verfügen.

Zum Austausch zwischen den Services dienen Hilfsklassen, die im Kapitel „Bestell-/Lieferorganisation“ beschrieben sind.

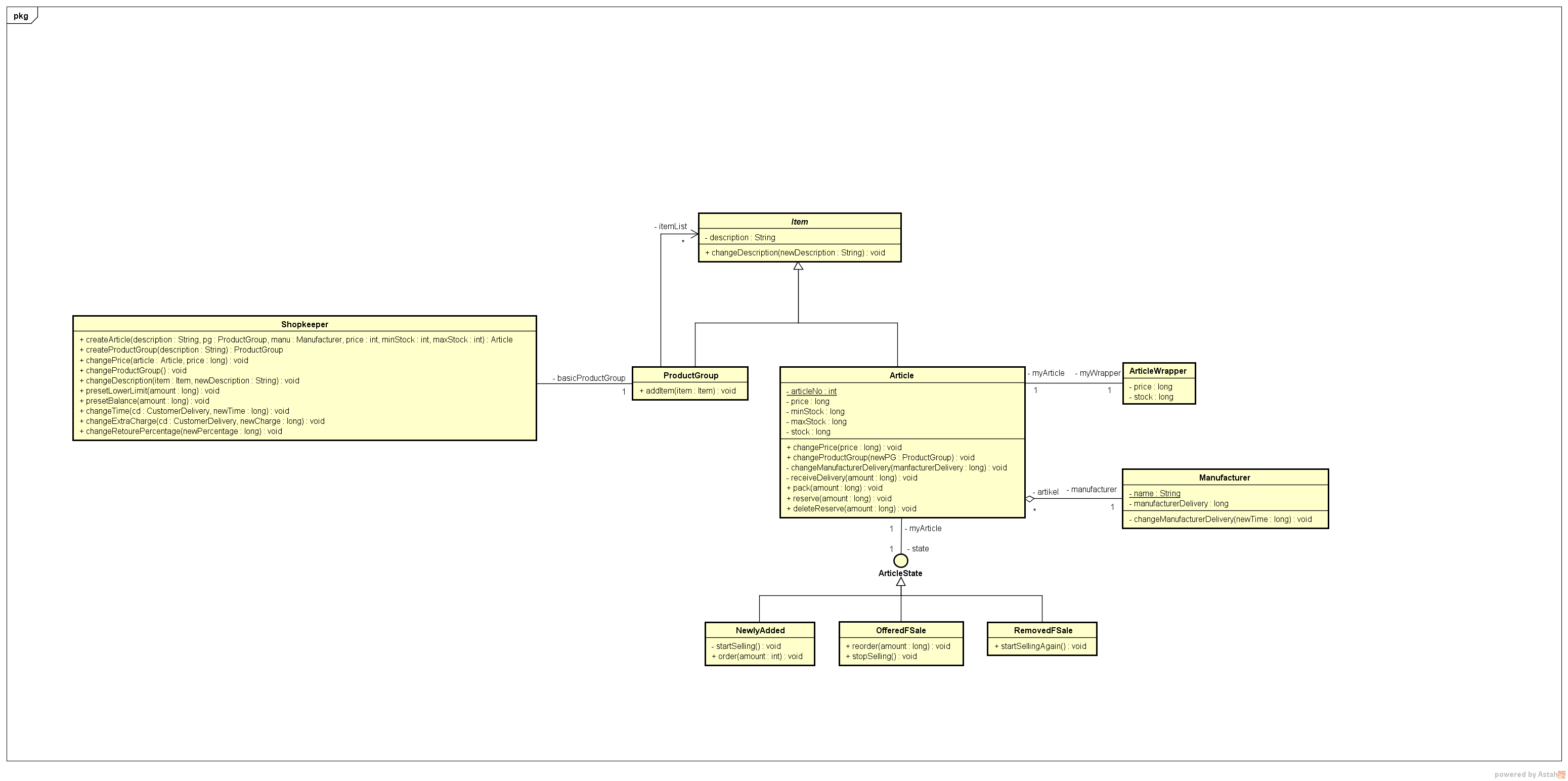
# Services

Die Architektur der Anwendung ist servicebasiert. Jede Sicht des Nutzers auf die Anwendung erfolgt über einen eigenen Service. Alle Services erben von der Klasse **Service** (siehe Klassenmodell).



Dem Ladenbesitzer steht der ShopkeeperService zur Verfügung und dem Kunden der ShopService, CustomerManager, AccountService sowie OrderService. Diese werden im Folgenden erläutert.

## Ladenbesitzer

Der Ladenbesitzer kann zentral auf die Lagerverwaltung zugreifen. Standardmäßig ist es der Benutzer „hugo“.

Das Lager bzw. der Ladenbetreiber-Service ist wie im folgenden Klassenmodell aufgebaut.

### Lager

Im Lager liegen alle Verkaufsgegenstände (Items). Es besitzt initial eine Produktgruppe namens „Produkte“, unter der alle weiteren Artikel und Produktgruppen angeordnet sind. (entspricht im Klassenmodell dem Shopkeeper)

### Gegenstände

Gegenstände (Items) sind abstrakte Verkaufseinheiten. Sie besitzen eine eindeutige Beschreibung. Implementierungen müssen ihre jeweilige Beschreibung ändern können sowie ihre Artikelanzahl zurückliefern können.

Gegenstände sind sowohl Artikel als auch Produktgruppen.

### Produktgruppen

Produktgruppen dienen zur Klassifikation und Kategorisierung von Artikeln. Sie besitzen einen eindeutigen Namen sowie eine Liste von Gegenständen.

Es besteht die Möglichkeit, über die Methode addItem(Item) einen Gegenstand zu der Produktgruppe hinzuzufügen.

### Artikel

Artikel sind konkrete Verkaufseinheiten. Sie besitzen jeweils einen Hersteller, eine Beschreibung, einen Mindest- sowie Maximalbestand, eine Herstellerlieferzeit sowie den momentanen Bestand. Ein Artikel ist immer unter einer Produktgruppe angeordnet, die beliebig gewechselt werden kann.

* **Beschreibung ändern –** changeDescription(String)
* **Artikelpreis ändern –** changePrice(Integer)
* **Herstellerlieferzeit ändern –** changeManuDelivery(Integer)
* **Produktgruppe wechseln –** changeProductGroup(ProductGroup)

Der Artikel kann sich in drei verschiedenen Zuständen befinden: der Artikel kann neu, im Verkauf oder aus dem Verkauf genommen sein. Statusspezifische Aktionen müssen von diesen direkt ausgeführt werden und werden nicht über den Article delegiert.

* **Neuer Artikel:** ein neuer Artikel kann nicht von Kunden bestellt werden. Dazu muss er in den Verkauf genommen werden, was durch die Methode startSelling(), die den Status auf OfferedFSale (Im Verkauf) setzt.
* **Im Verkauf:** Artikel, die im Verkauf sind, werden automatisch nachbestellt, wenn ihr momentaner Lagerbestand unter den Mindestlagerbestand fällt. Das Bestellverfahren entspricht dem Bestellpunktverfahren der verbrauchsgesteuerten Disposition. Mindest- sowie Maximalbestand sind also als dessen untere bzw. obere Grenze zu betrachten. Zur Beobachtung dient ein eigener Thread (common.ArticleStockChecker, nicht im Klassenmodell enthalten), der die Lagerbestände aller Artikel jede Sekunde kontrolliert. Und entsprechende Bestellungen auslöst.  
  Soll der Artikel nicht mehr nachbestellt werden, muss die Funktion stopSelling() ausgeführt werden, die den Artikel in einen RemovedFSale versetzt.  
  OfferedFSale bietet weiterhin die Funktion addToCart(...), mit Hilfe derer ein Artikel zum Einkaufswagen hinzugefügt wird.
* **Aus dem Verkauf genommen:** Artikel, die aus dem Verkauf genommen sind, werden zwar noch verkauft aber nicht mehr nachbestellt. Es können also vom Kunden auch keine Vorbestellungen ausgelöst werden. Die Methode StartSellingAgain() dagegen wechselt den Status wieder zurück auf OfferedFSale.

### Kundenkonten/Lieferbedingungen

Dem Ladenbesitzer stehen Möglichkeiten zur Verfügung, Kundenkonten zu prekonfigurieren.

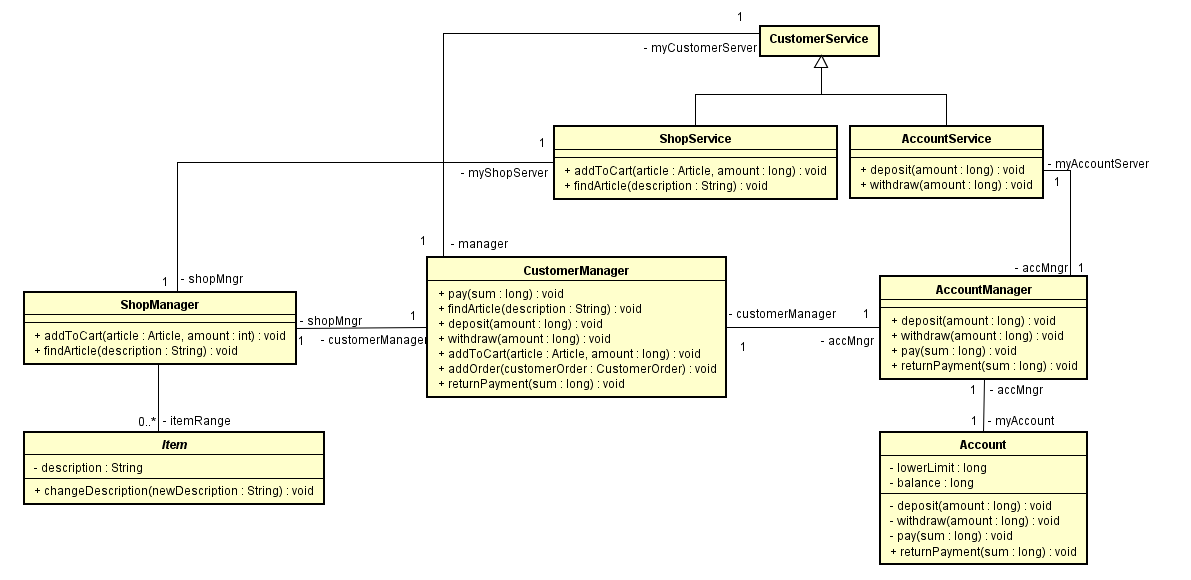
In der Klasse serverConstants.ConfigConstants existieren statische Variablen, in denen Informationen zur Retourengebühr, des voreingestellten Kontostands sowie des unteren Limits neuer Kundenkonten zur Laufzeit abgelegt sind. Die Variablen können auf diese Weise klassenübergreifend ausgelesen und verändert werden.

Die dem Ladenbesitzer zur Verfügung stehenden Operationen in diesem Sinne editieren jene Attributwerte. Die Kundenkonten lesen die Werte PresetAccountBalance und PresetAccountLowerLimit einmalig bei Anlage aus und die RetourePercentage bei jedem Retourenvorgang.

## Kunde

Der Kunde muss sich vor Nutzung des Systems am Server registrieren. Dazu dient der Registerservice.

Anschließend kann sich der Kunde mit den gewählten Anmeldenamen und Passwort anmelden. Dadurch erhält er Zugriff auf vier weitere Services, die alle von Spezialisierungen von CustomerService sind. Damit die verschiedenen Manager miteinander kommunizieren können, wird die Klasse CustomerManager als Schnittstelle verwendet.

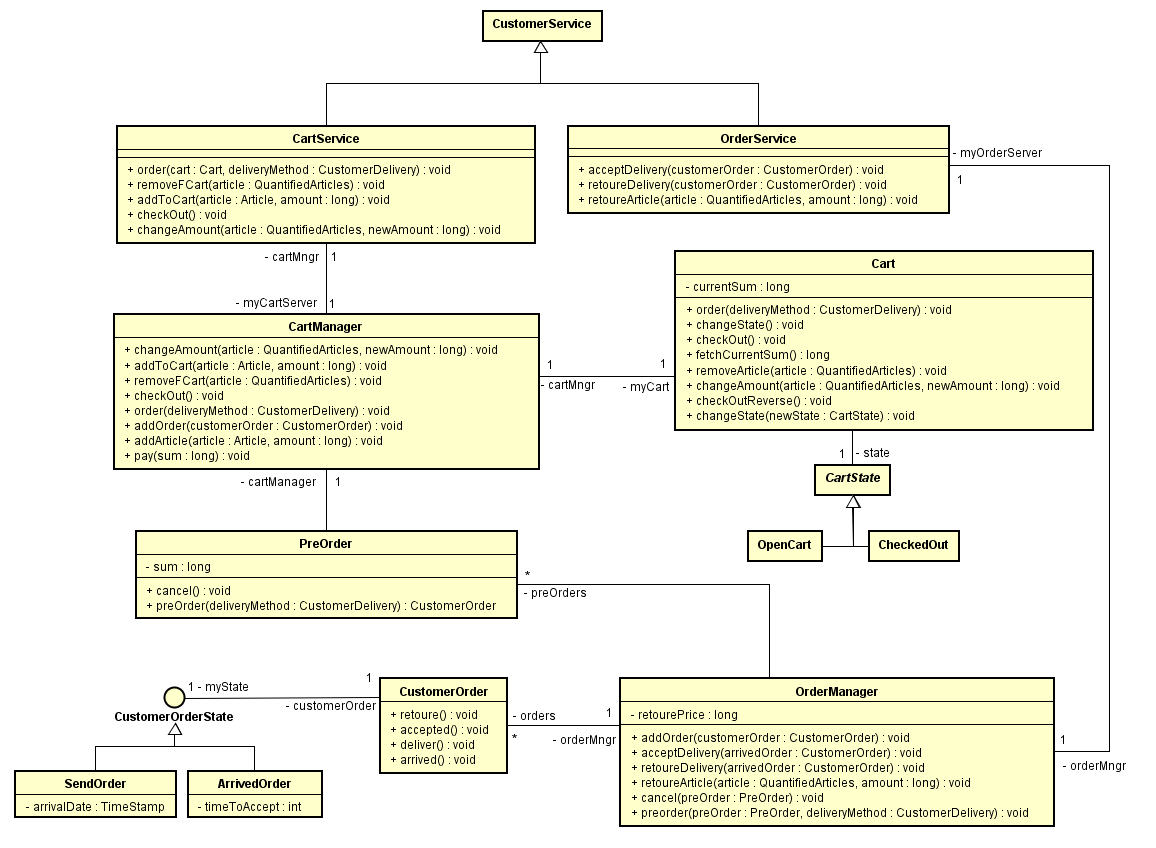


### Shop und Account

Im Shop liegen ähnlich wie im Lager alle Verkaufsgegenstände. Im Gegensatz zum Lager sind hier aber nur Preis, Hersteller, Beschreibung, Lagerbestand und Zustand zu sehen. Es gibt die Möglichkeit Artikel anhand ihrer Beschreibung zu suchen.

Der Account handhabt alle Änderungen, die ein Kunde an seinem Account vornehmen kann. Dazu gehört neben dem Einzahlen von Geld auch das Abheben von selbigem. In den Methoden dieser Klasse wird gewährleistet, dass das untere Limit stets eingehalten und nicht unterschritten wird.

### Einkaufswagen



Der Einkaufswagen ist zusammen mit dem OrderManager, die zentralen Elemente bei dem Bestellmanagement. Während im Einkaufswagen alle Artikel, die im Shop ausgewählt wurden, aufgelistet sind, enthält der OrderManager alle Bestellungen, die aktuell unterwegs oder geliefert sind. Retournierte und akzeptierte Bestellungen sind nicht mehr sichtbar.

Der Einkaufswagen kann einen von zwei Zuständen annehmen: ein neu erstellten Einkaufswagen, der noch nicht im Bestellvorgang ist, oder ein zur Kasse gebrachter Einkaufswagen, bei dem die enthaltenen Artikel bereits aus dem Lager genommen wurden, allerdings noch nicht verschickt.

* **Neu erstellter Einkaufswagen**: in diesem Zustand ist der Einkaufswagen von Anfang an. Durch die Methode checkOut() (Zur Kasse gehen) wird er in den Zustand CheckedOut gesetzt.
* **Zur Kasse gebrachter Einkaufswagen**: Nur in diesem Zustand kann ein Einkaufswagen entgültig bestellt werden. Werden nach einem Wechsel in diesem Zustand Änderungen am Einkaufswagen vorgenommen geht er zurück in den OpenCart-Zustand und muss erneut ausgecheckt werden.

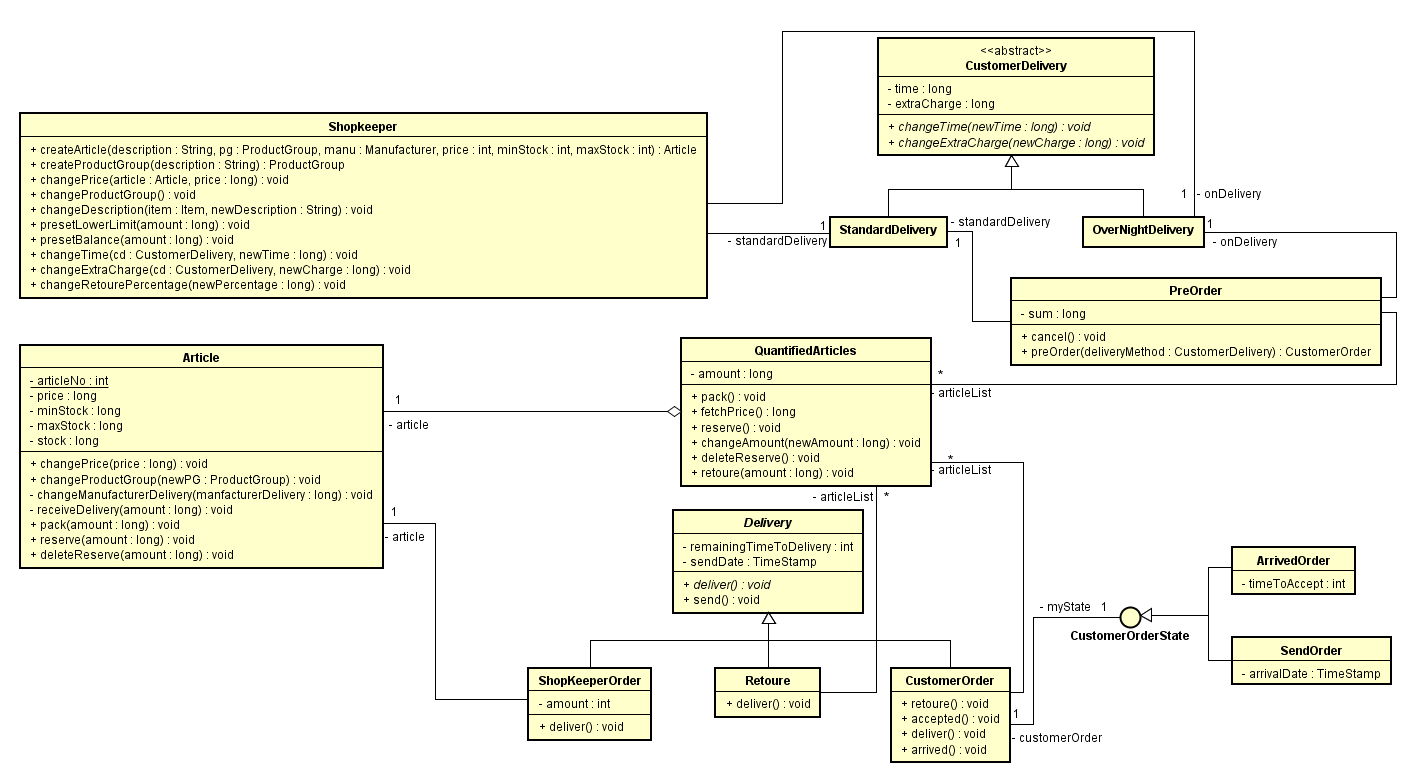
### Bestellungen

Bestellungen sind in drei Varianten aufgeteilt: ShopkeeperOrder, CustomerOrder und Retoure. Die CustomerOrder verfügt im Gegensatz zu den beiden anderen über einen Zustand und ist im OrderManager sichtbar. Die Zustände der CustomerOrder sind SendOrder, solange sich die Bestellung abgesendet, aber noch nicht angekommen ist und ArrivedOrder, sobald die Bestellung angekommen ist.

Nur eine Sendung, die sich im Zustand ArrivedOrder befindet, kann angenommen oder retourniert werden.

# Bestell-/Lieferorganisation

Die Bestellung- und Lieferorganisation erfolgt über nebenläufiges Ausführen von Tasks, die in einem TaskManager durchgeführt werden. Da diese aber nicht zum eigentlichen Ablauf gehören sondern lediglich die Nebenläufigkeit erzeugen sind sie im Klassenmodell nicht aufgelistet.



## Quantifizierte Artikel

QuantifiedArticles ist eine Hilfsklasse, die Posten einer Bestellung repräsentieren. Sie beziehen sich immer auf genau einen Artikel und enthalten einen amount, der angibt wie oft der referenzierte Artikel von einem Objekt dieser Klasse repräsentiert wird. Sowohl CustomerOrder als auch Retoure nutzen eine Liste von Objekten dieser Klasse um festzulegen, was die Bestellung enthält. ShopkeeperOrder bezieht sich direkt auf den Artikel und nutzt QuantifiedArticles nicht.

## Kundenlieferzeiten

Die Kundenlieferzeiten sind über eine abstrakte Klasse CustomerDelivery realisiert. Diese dient als Abstraktion der eigentlichen Kundenlieferzeiten, welche als Singletons realisiert wurden. Die Kundenlieferzeiten können im Shopkeeper geändert werden. Währenddessen nutzen alle Bestellungen die Kundenlieferzeiten, um die Verzögerung der nebenläufig durchgeführten Aktionen zu bestimmen.

### Bestellungen

Delivery ist eine abstrakte Klasse, die als Vorlage für alle Bestellungen dient. Alle Bestellungen greifen auf die zuvor beschriebenen nebenläufigen Tasks zu, wenn sie versendet (send) werden, allerdings werden abhängig von der Art der Bestellung bei Ankunft (deliver) andere Aktionen durchgeführt. Während bei ShopkeeperOrder und Retoure die Annahme automatisch durch die Artikel selbst erfolgt, müssen CustomerOrder aktiv vom Kunden über die Oberfläche akzeptiert werden (accepted). Sollte dies nicht in der vorgegebenen Zeit erfolgen, wird die gesamte CustomerOrder als Retoure zurückgesandt.

### Vorbestellungen

Eine Vorbestellung wird automatisch erzeugt, wenn ein Einkaufswagen ausgecheckt werden soll, für den nicht genug Artikel im Lager sind. Diese Vorbestellung prüft bei jeder Änderung des Lagerbestands seiner Artikel, ob genug Artikel vorhanden sind und reserviert diese gegebenenfalls. Sobald eine PreOrder ihre Artikel reserviert hat, kann sie bestellt werden, wodurch sie zu einer normalen Bestellung wird. Eine PreOrder kann jederzeit storniert werden, wodurch eventuell reservierte Artikel wieder freigegeben werden.