WS2010/2011
BAI3-SEP1 Gruppe 2
Aufgabenblatt 1 Aufgabe 2

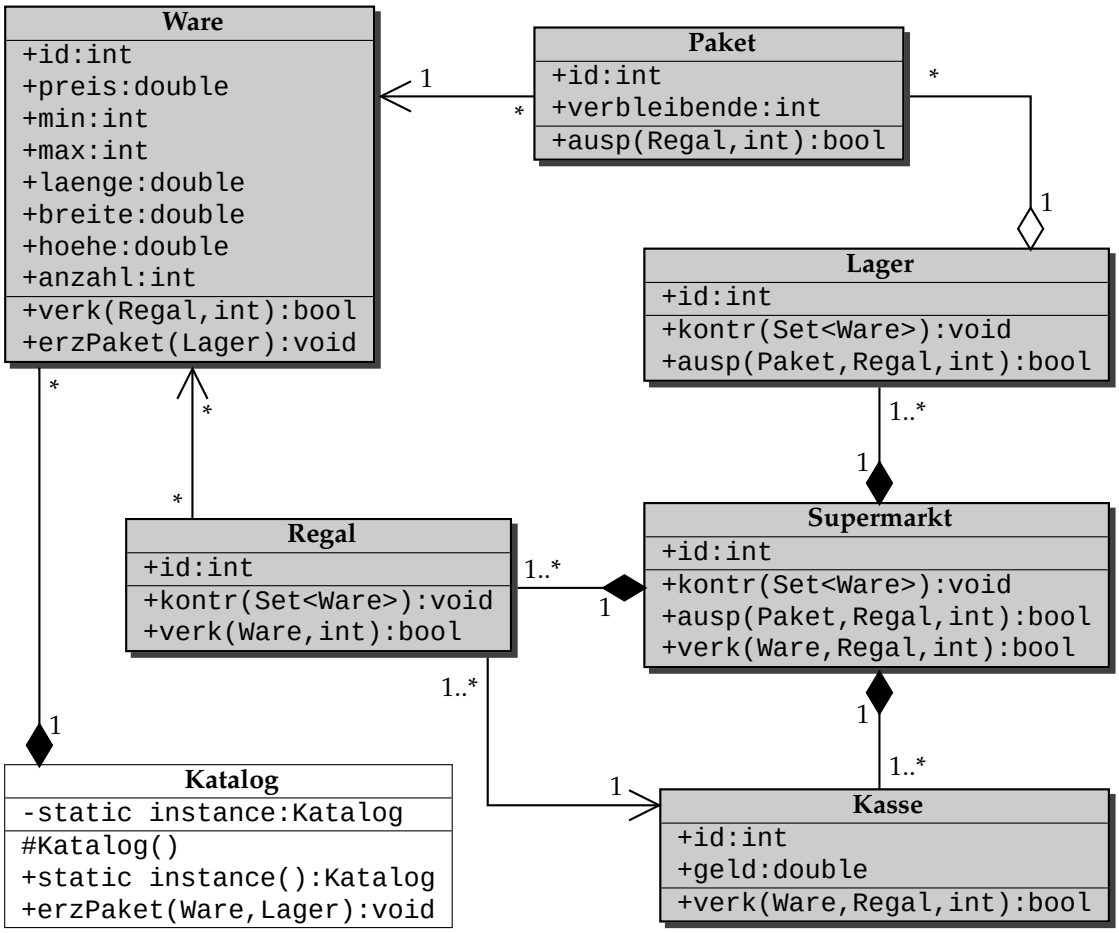
R. C. Ladiges, D. Fast

12. November 2010

Inhaltsverzeichnis

2 Aufgabe 2

2.1 UML-Diagramm



Methodennamen aus Platzgründen abgekürzt.

2.2 Textuelle Beschreibung

Auf Getter- und Setter-Methoden wird der Einfachheit halber verzichtet. In einer Implementation wäre dies, zusammen mit den passenden Sichtbarkeiten der Methoden und Instanzvariablen, hinzuzufügen.

2.2.1 Supermarkt

Dieses Modell unterstützt die Existenz von mehreren *Supermärkten*. Diese *Supermarkt*-Objekte haben unabhängig voneinander mehrere *Lager*, *Kassen* und *Regale*. Um diese referenzieren zu können, haben *Supermärkte* Mengen von diesen Objekten als Instanzvariablen.

Attribute:

- **`Set<Lager> lager`**
  Menge der *Lager* die zu diesem *Supermarkt* gehören.
- **`Set<Regal> regale`**
  Menge der *Regale* die zu diesem *Supermarkt* gehören.
- **`Set<Kassen> kassen`**
  Menge der *Kassen* die zu diesem *Supermarkt* gehören.

Methoden:

- **`void kontrollieren(Set<Ware>)`**
  um in allen *Lagern* und *Regalen* den Bestand der angegebenen *Waren* zu überprüfen, und gegebenenfalls über **`auspacken`** neue aus dem *Lager* zu holen oder neue *Pakete* zu bestellen.
- **`bool auspacken(Paket, Regal, int)`**
  um eine bestimmte Anzahl *Waren* aus einem *Paket* auszupacken und in ein *Regal* zu stellen. ***return false***, wenn die verbleibende Anzahl im *Paket* nicht ausreicht, das *Regal* danach zu viele Artikel dieser *Ware* hätte, das *Paket* nicht in einem *Lager* dieses *Supermarktes* ist, oder das *Regal* nicht in diesem *Supermarkt* ist.
- **`bool verkaufen(Ware, Regal, int)`**
  um eine bestimmte Anzahl *Waren* aus einem *Regal* zu entnehmen und an der zum *Regal* gehörenden *Kasse* zu verkaufen. ***return false***, wenn die Anzahl *Waren* im *Regal* nicht ausreicht, oder das *Regal* nicht in diesem *Supermarkt* ist.

2.2.2 Katalog

Unabhängig von den einzelnen *Supermärkten* gibt es einen *Katalog*. In diesem *Katalog* sind alle *Waren* für alle *Supermärkte* in einer Menge aufgelistet. Er ist ein Singleton[1] und ist dementsprechend im ganzem System einzigartig und bekannt. Mit ihm können die verschiedenen *Supermärkte* an einer *globalen* Stelle *Waren* finden und bestellen.

Attribute:

- **`private static final Katalog instance`**
  Die versteckte Klassenkonstante für das einzige *Katalog*-Objekt.
- **`Set<Ware> sortiment`**
  Menge von *Waren* im System.

Methoden:

- **`protected Katalog()`**
  Der Konstruktor wird geschützt, um zu verhindern dass weitere *Katalog*-Objekte erstellt werden können.
- **`public static Katalog instance()`**
  Die öffentliche Methode um das einzige *Katalog*-Objekt zurückzugeben (und gegebenenfalls zu erstellen).
- **`void erzeugePaket(Ware, Lager)`**
  um neue *Paket*-Objekte einer bestimmten *Ware* zu erstellen und in ein *Lager* zu bringen.

[1] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Design Patterns. Elements of Reusable Object-Oriented Software*, Seite 127 - `http://en.wikipedia.org/wiki/Singleton_pattern`

2.2.3 Ware

Ware ist eine Klasse, dessen Objekte unterschiedliche *Waren* im System darstellen. Diese können in jedem *Supermarkt* des Systems angeboten werden. *Waren* sind lediglich eine Beschreibung von einem Artikeltyp, und keine spezifische physische Objekte. Deshalb stehen sie auch im *Katalog* und werden von *Regalen* und *Paketen* lediglich referenziert.

Attribute:

- **`double preis`**
  der Preis in Euro die ein Artikel dieser *Ware* kostet.
- **`int min`**
  die minimale Anzahl Artikel dieser *Ware* die in einem *Regal* stehen sollte.
- **`int max`**
  die maximale Anzahl Artikel dieser *Ware* die in einem *Regal* stehen sollte.
- **`double laenge`**
  Länge eines *Paketes* dieser *Ware*.
- **`double breite`**
  Breite eines *Paketes* dieser *Ware*.
- **`double hoehe`**
  Höhe eines *Paketes* dieser *Ware*.
- **`int anzahl`**
  Anzahl Artikel die in ein *Paket* dieser *Ware* passen.

Denkbar wären auch weitere Attribute, wie z.B.: Bezeichnung, Hersteller, Haltbarkeitsdatum, Mindest-/Maximalanzahl im Lager, etc.

Methoden:

- **`bool verkaufen(Regal, int)`**
  um eine bestimmte Anzahl dieser *Ware* aus einem *Regal* zu entnehmen und an der zum *Regal* gehörenden *Kasse* zu verkaufen. ***return false***, wenn die Anzahl *Waren* im *Regal* nicht ausreicht.
- **`void erzeugePaket(Lager)`**
  um neue *Paket*-Objekte dieser *Ware* zu erstellen und in ein *Lager* zu bringen.

Die **`verkaufen`**-Methode ist in dieser Klasse, damit ein Anwender der Software, der z.B. durch die *Waren* eines *Regals* iteriert und ein *Waren*-Objekt bekommt, direkt mit diesem Objekt die Methode aufrufen kann, um *Waren* zu verkaufen. Auch wenn dies zwar in der Realität etwas anders aussieht (verkauft wird an der *Kasse*), hat man im System eine höhere Flexibilität.

2.2.4 Paket

Paket-Objekte sind einzigartige physische Objekte (andere *Pakete* der gleichen *Ware* sind ein anderes Objekt). Jedes *Paket* einer *Ware* enthält eine Referenz auf die *Ware* die in ihr gelagert ist, und speichert die verbleibende Anzahl der im *Paket* vorhandenen Artikel dieser *Ware*.

Attribute:

- `Ware ware`
  Referenz auf die *Ware* die in dem *Paket* ist.
- `int verbleibende`
  Anzahl der verbleibenden Artikel im *Paket*.

Methoden:

- `bool auspacken(Regal, int)`
  um eine bestimmte Anzahl Artikel aus diesem *Paket* auszupacken und in ein *Regal* zu stellen. ***return false***, wenn die verbleibende Anzahl nicht ausreicht, das *Regal* danach zu viele Artikel dieser Ware hätte, oder das *Regal* nicht in diesem *Supermarkt* ist.

2.2.5 Lager

Im *Lager* eines *Supermarktes* werden *Pakete* gelagert. Es kann mehrere *Lager* in einem *Supermarkt* geben.

Attribute:

- `Set<Paket> lager`
  Menge der *Pakete* in diesem *Lager*.

Methoden:

- `void kontrollieren(Set<Ware>)`
  um im *Lager* den Bestand der angegebenen *Waren* zu überprüfen, und gegebenenfalls neue *Pakete* zu bestellen.
- `bool auspacken(Paket, Regal, int)`
  um eine bestimmte Anzahl *Waren* aus einem *Paket* dieses *Lagers* auszupacken und in ein *Regal* zu stellen. ***return false***, wenn die verbleibende Anzahl im *Paket* nicht ausreicht, das *Regal* danach zu viele Artikel dieser *Ware* hätte, das *Paket* nicht in diesem *Lager* ist, oder das *Regal* nicht in diesem *Supermarkt* ist.

2.2.6 Regal

Ein *Regal* ist ein Objekt in dem *Waren* in einer bestimmten Anzahl ausgestellt werden. Jedes *Regal* hat eine ihr zugewiesene *Kasse*, an der die Artikel des *Regals* verkauft werden.

Attribute:

- `Kasse kasse`
  Referenz auf die dem *Regal* zugewiesene *Kasse*.
- `Map<Ware,Integer> waren`
  Bag[2] von *Waren*, die in diesem *Regal* zum Verkauf angeboten werden.

[2]Multiset, eine Menge in der die Elemente mehrfach vorkommen können. Implementiert als Map mit der *Ware* als Key und der Anzahl als Value.
`http://en.wikipedia.org/wiki/Multiset - http://en.wikipedia.org/wiki/Map_(mathematics)`

Methoden:

- **`void kontrollieren(Set<Ware>)`**
  um in diesem *Regal* den Bestand der angegebenen *Waren* zu überprüfen, und gegebenenfalls über **`auspacken`** neue aus dem *Lager* zu holen.
- **`bool verkaufen(Ware, int)`**
  um eine bestimmte Anzahl *Waren* aus diesem *Regal* zu entnehmen, und an der zu diesem *Regal* gehörenden *Kasse* zu verkaufen. ***return false***, wenn die Anzahl *Waren* im *Regal* nicht ausreicht.

Die **`verkaufen`**-Methode ist in dieser Klasse, damit ein Anwender der Software, der z.B. durch die *Waren* dieses *Regals* iteriert, die *Waren*-Objekt die er bekommt direkt an diese Methode geben kann, um diese *Waren* aus diesem Regal zu verkaufen. Auch wenn dies zwar in der Realität etwas anders aussieht (verkauft wird an der *Kasse*), hat man im System eine höhere Flexibilität.

2.2.7 Kasse

An einer *Kasse* werden *Waren* verkauft, und der geldliche Erlös gesammelt.

Attribute:

- **`double geld`**
  Einnahmen in Euro. Wird von der **`verkaufen`**-Methode mit dem Einzelpreis der *Ware* multipliziert mit der Anzahl Artikel aufaddiert.

Methoden:

- **`bool verkaufen(Ware, Regal, int)`**
  um eine bestimmte Anzahl *Waren* aus einem *Regal* zu entnehmen und an dieser *Kasse* zu verkaufen.***return false***, wenn die Anzahl *Waren* im *Regal* nicht ausreicht, oder diese *Kasse* nicht zu dem *Regal* gehört.

Informationen zur Signatur

	**Unterzeichner**	EMAILADDRESS=robin.ladiges@haw-hamburg.de, CN=Robin Christopher Ladiges
	Datum/Zeit	Fri Nov 12 11:52:20 CET 2010
	Austeller-Zertifikat	CN=CAcert Class 3 Root, OU=http://www.CAcert.org, O=CAcert Inc.
	Serien-Nr.	44727
	Methode	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signatur)