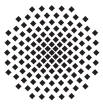

Universität Stuttgart

SQA
Software Quality and Architecture Group

Übungsblatt 10

Programmierung und Softwareentwicklung (WS 2024/2025)

Abgabe: Fr. 17.01.2025, 23:59 Uhr — Besprechung: KW 03

- Bitte lösen Sie die Übungsaufgabe in **Gruppen von 2 Studierenden**.
- Dieses Übungsblatt besteht aus zwei Teilen (A, B). Teil A ist in der Präsenzübung zu lösen. Teil B ist in Heimarbeit (Gruppe von 2 Studierenden) zu lösen und rechtzeitig abzugeben. Die Abgabe erfolgt über ILIAS.
- Geben Sie `.java`-Dateien nur im UTF-8 Encoding ab. Ändern Sie das Textdateiencoding auf UTF-8 ab, bevor Sie die Unterlagen herunterladen. Abhängig von Ihrem Betriebssystem müssen Sie möglicherweise auch nichts tun.
- Geben Sie zu Beginn der Dateien Ihre Namen (Vor- und Nachname), die Matrikelnummern und die E-Mail-Adressen an. Nutzen Sie bei Java-Dateien die korrekte JavaDoc-Syntax.
- Benennen Sie die Dateien nach dem folgenden Schema:
  1. **PSE**[**ÜB-Nr**]-[**Nachnamen der Teammitglieder**]-[**Nachname des Tutors**]**.pdf**
     Beispiel: PSE10-StießSpethKrieger-Becker.pdf
  2. **PSE**[**ÜB-Nr**]-[**Nachnamen der Teammitglieder**]-[**Nachname des Tutors**]**.zip**
     Beispiel: PSE10-StießSpethKrieger-Becker.zip
- Missachtung der formalen Kriterien kann dazu führen, dass einzelne Aufgaben oder die gesamte Abgabe mit 0 Punkten bewertet werden.

Lernziel: Auf diesem Übungsblatt werden Sie Ihr Wissen über Modellierung vertiefen. Des Weiteren werden Sie den Umgang mit OOP und Collections weiter üben.

Punkte: Dieses Übungsblatt enthält zwei Teile. In Teil B können Sie bis zu 60 Punkte und 4 Bonuspunkte erzielen. Zum Bestehen des Blatts benötigen Sie mindestens 30.0 Punkte.

Style: Bitte halten Sie die in der Vorlesung vorgestellten Style-Regeln ein. Dazu gehören auch JavaDoc sowie Vor- und Nachbedingungen. Der Stil Ihrer Implementierung wird mit bis zu 50% bewertet.

Vorbereitung: Bitte erledigen Sie die folgenden Schritte **vor** der Präsenzübung.

- Importieren Sie das zu diesem Übungsblatt gehörende Maven Projekt in Ihre IDE. Sie finden das Maven Projekt in unserem Git-Repository:
  `https://github.com/SQA-PSE-WS-2024-2025/exercise-sheet-10`
- Stellen Sie sicher, dass Sie Übungsblatt 09 absolviert haben, sowie alle Software installiert und funktionsfähig ist (IDE (Eclipse, IntelliJ, VSCode,...) und Java 21).
- Machen Sie sich mit einem der unter *Unterlagen* genannten Tools zum Modellieren von UML Diagrammen vertraut.

Unterlagen:

- Git-Repositories: `https://github.com/SQA-PSE-WS-2024-2025/`
- Dokumentation des Hamstersimulators: `https://tinyurl.com/5yx654w8`
- Tools zum Modellieren von UML Diagrammen: `https://mermaid.js.org/`, `https://app.diagrams.net/`, `https://hylimo.github.io/` (nur Klassendiagramme)

Scheinkriterien: Durch die Teilnahme am Übungsbetrieb können Sie sich für die Teilnahme an der Klausur qualifizieren:

- Bestehen von min. 80% aller Übungsblätter.
- Ein Übungsblatt gilt als bestanden, wenn 50% der Punkte des abgegebenen Heimarbeitsteils erreicht wurden.
- Aktive Teilnahme an min. 80% der Übungen.

Viel Erfolg!

1 Teil A - Präsenzaufgaben

Aufgabe 1 UML Klassendiagramme I

Ein Kommilitone erzählt Ihnen von seiner neuen App-Idee. In der App mit dem Namen PummelmonFly züchten Forscher genmanipulierte Kampffliegen und lassen sie gegeneinander in Arenenkämpfen antreten. Da Ihr Kommilitone nie die PSE-Veranstaltung besucht hat, bittet er Sie, den Entwurf zu übernehmen und ein Domänenmodell[1] zu erstellen. In einem für den nächsten Tag angesetzten Interview erfahren Sie folgende weitere Informationen:

- Jeder Forscher besitzt einen Namen, eine Anzahl an Erfahrungspunkten und einen FlyDex.
- Zusätzlich kann jeder Forscher bis zu 5 Kampffliegen besitzen.
- Generell können drei Gruppen von Kampffliegen unterschieden werden: DickeBrummer, SpeedFlys und MediCopter.
- Jede Fliege hat dabei einen Namen, eine Geschwindigkeit und besitzt einen Standardangriff.
- Standardangriffe besitzen wiederum einen Namen und einen bestimmten Grad an Schaden, den sie beim Gegner hervorrufen.
- Im FlyDex jedes Forschers sind die Kampffliegen aufgeführt, die der Forscher gesehen hat.

Zusätzlich zeigt Ihnen Ihr Kommilitone einen ersten Prototypen der Arena:

```java
public class Arena {

    private final Researcher attacker;
    private final Researcher defender;
    private final Reward prize;

    public void fight() {...};

    // getter and setter
}
```

Listing 1: Code des Arena Prototyps

(a) Was gehört in ein Domänenmodell und was nicht? Nennen Sie Beispiele aus der PummelmonFly Domäne. Diskutieren Sie diese Frage mit Ihrem Nachbarn und schreiben Sie Ihre Ergebnisse auf.

(b) Welche UML Diagramme eignen sich besonders, um ein Domänenmodell zu entwerfen?

(c) Worin unterscheiden sich die Klassen der verhaltensorientierten und der strukturorientierten Diagramme in der UML? Was stellen Sie dar? Geben Sie für jede Klasse ein Beispiel.

(d) Worin unterscheiden sich Aggregation und Komposition? Nennen Sie jeweils ein Beispiel.

(e) Erstellen Sie unter Berücksichtigung aller Ihnen vorliegenden Informationen ein UML Klassendiagramm, das den PummelmonFly Sachverhalt möglichst vollständig darstellt.

Hinweis: Denken Sie sich keine zusätzlichen Informationen aus. Berücksichtigen Sie das Prinzip des Information Hidings.

Hinweis: Sie dürfen gerne Tools zum Modellieren des UML Diagramms verwenden, z.B. eines der auf dem Deckblatt genannten Tools.

Hinweis: Achten Sie auch eine Korrekte und vollständige Notation.

[1] Also ein Modell, dass unsere *Domäne* abbildet. Hier ist die Domäne PummelmonFly.

Aufgabe 2 UML Klassendiagramme II

Die folgende Aufgabe befasst sich mit UML-Klassendiagrammen. Dazu finden Sie in Abbildung 1 die Struktur eines Verwaltungsprogramms für Ställe.

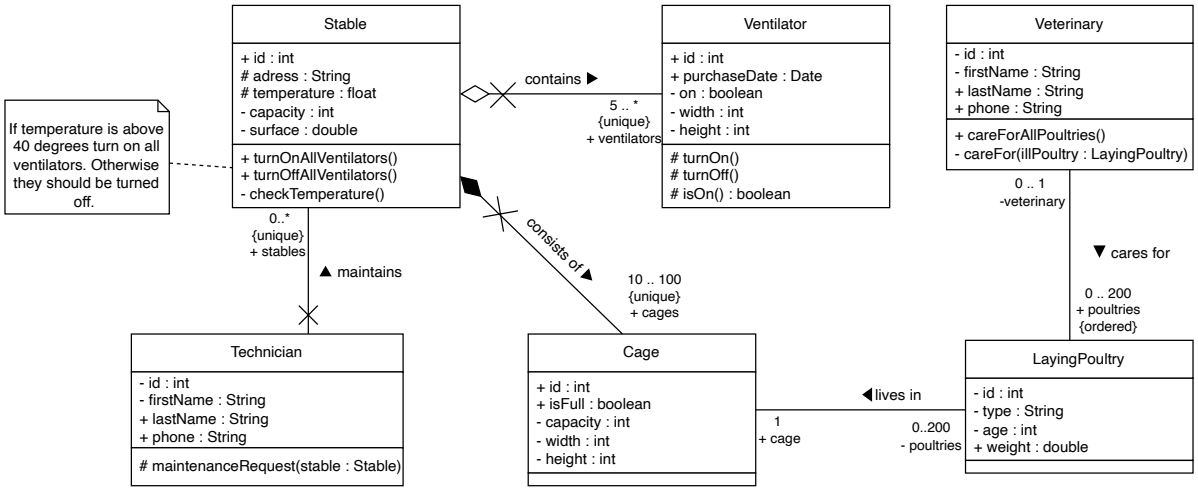

Abbildung 1: UML-Klassendiagramm des Stalls

(a) Beschreiben Sie die Beziehung zwischen einem Stall (Stable) und einem Käfig (Cage) und erläutern Sie dabei die Semantik. Schreiben Sie Ihre Ergebnisse auf.

(b) Im Paket `de.unistuttgart.iste.sqa.pse.sheet10.presence.stables` finden Sie die Implementierung des obigen UML-Diagramms. Der Code und das Modell sind an einigen Stellen inkonsistent.

   i. Zum Beispiel hat die Klasse `Veterinary` im Model die Operationen `careForAllPoultries()` und `careFor(LayingPoultry illPoultry)`, die im Code beide fehlen. Verbessern Sie diese Inkonsistenz, indem Sie die fehlenden Operationen im Code ergänzen.

   ii. Finden und verbessern Sie mindestens zwei weiter Inkonsistenzen im Code oder im Modell.

(c) Beschreiben Sie eine Erweiterung des Klassendiagramms, die es ermöglicht, dass in einem Käfig andere Arten von Geflügeln leben können (z.B. MeatPoultry, etc.), die Klasse `LayingPoultry` aber bestehen bleibt. Beachten Sie dabei die Grundsätze der OOP.

(d) Setzen Sie Ihre Erweiterung aus der vorherigen Frage in Java um. Achten Sie darauf, dass Sie alle betroffenen Stellen anpassen.

2 Teil B - Heimarbeit

Aufgabe 1 Polymorphie (PDF)

In dieser Aufgabe beschäftigen Sie sich mit der Korrektheit von Vererbung.

Im Paket `homework.oop` befinden sich die Klassen `LazyHamster` und `CheatingHamster`, die jeweils das Kommando `move` der Klasse `Hamster` überschreiben. In Listing 3 finden Sie die Implementierung des Befehls `move` der Klasse `Hamster` inklusive Vor- und Nachbedingungen.

Hinweis: In Listing 3 sind die Vor- und Nachbedingungen der Operation in formal korrektem JML notiert. Konsultieren Sie den Abschnitt *Veträge* im Foliensatz *03 - Schnittstellen* falls Sie nicht (mehr) wissen, wie man JML liest.

Hinweis: Diese Aufgabe ist eine reine Textaufgabe. Falls Sie für die Teilaufgaben (a) oder (b) Dokumentation in den vorgegebenen Klassen erstellen, so kopieren Sie diese am Ende bitte mit in die PDF. Für diese Aufgabe werden *nur* Lösungen im PDF bewertet.

Hinweis: Falls Sie das Verhalten eines faulen oder betrügerischen Hamster ausprobieren wollen, können Sie dazu die Klasse `JustSomeHamsterGame` aus dem Paket `homework` verwenden. Auch hier gilt: Lösungen am Ende bitte in die PDF kopieren.

(a) *(3 Punkte)* Beschreiben Sie, was die Operation `move` in der Klasse `LazyHamster` tut. Beschreiben Sie außerdem die Vor- und Nachbedingungen der Operation in natürlicher Sprache.

(b) *(3 Punkte)* Beschreiben Sie, was die Operation `move` in der Klasse `CheatingHamster` tut. Beschreiben Sie außerdem die Vor- und Nachbedingungen der Operation in natürlicher Sprache.

(c) *(4 Punkte)* Vergleichen Sie die von Ihnen formulierten Vor- und Nachbedingungen der `move` Operationen aus den beiden Kindklassen mit den Vor- und Nachbedingungen der Elternklasse (siehe Listing 3). Begründen Sie, warum die Implementierungen der `move` Operationen aus den beiden Kindklassen im Sinne der Vererbung *nicht* korrekt ist.

(d) *(4 Bonuspunkte)* Geben Sie einen Code-Schnipsel[2] an, bei dem die `move` Operation der Klasse `LazyHamster` oder der Klasse `CheatingHamster` zu Problemen führen könnte. Begründen Sie, warum Sie diesen Code-Schnipsel gewählt haben, und was das Problem ist.

```
1  /*@
2   @ requires frontIsClear();
3   @ requires isInitialized;
4   @ ensures getDirection() == Direction.NORTH ==>
5   @       \old(getLocation()).getRow() == getLocation().getRow() + 1 &&
6   @       \old(getLocation()).getColumn() == getLocation().getColumn();
7   @ ensures getDirection() == Direction.SOUTH ==>
8   @       \old(getLocation()).getRow() == getLocation().getRow() - 1 &&
9   @       \old(getLocation()).getColumn() == getLocation().getColumn();
10  @ ensures getDirection() == Direction.EAST ==>
11  @       \old(getLocation()).getRow() == getLocation().getRow() &&
12  @       \old(getLocation()).getColumn() == getLocation().getColumn() - 1;
13  @ ensures getDirection() == Direction.WEST ==>
14  @       \old(getLocation()).getRow() == getLocation().getRow() &&
15  @       \old(getLocation()).getColumn() == getLocation().getColumn() + 1;
16  @*/
17  /**
18   * Move the hamster one step towards its looking direction.
19   * @throws FrontBlockedException When the tile in front of the
20   * hamster is blocked
21   */
22  public void move() {
23      this.game.processCommandSpecification(
24              new MoveCommandSpecification(this.internalHamster));
25  }
```

Listing 3: Implementierung des Befehls `move` in `Hamster`.

[2] „Schnipsel“ heißt hier, dass Sie kein vollständige Operation oder gar Klasse angeben sollen, sondern nur wenige Zeilen, die z.B. innerhalb einer Operation vorkommen könnten.

Aufgabe 2 Versandhaus (`.zip`, PDF)

Hinweise zur Abgabe:
Als Abgabe dient in dieser Aufgbae komplette Projekt als .zip. Als Hilfestellung finden Sie eine Anleitung in *Anhang - Maven Projekte erstellen und exportieren*, in den Unterpunkten 3.2.2, 3.1.2 und 3.3.2.
Bitte beachten Sie die folgenden Punkte:

- Überprüfen Sie unbedingt, dass Sie das Projekt korrekt exportiert haben. Tun Sie dies, indem einer Ihrer Teampartner das .zip-Projekt bei sich importiert und öffnet.
- Achten Sie darauf, dass das Projekt alle für die Lösung des Heimarbeitsteil relevanten Java Dateien enthält. Falls Ihr Projekt außerdem die Java Dateien des Präsenzteils enthält, ist dies kein Fehler. Wir werden die Dateien des Präsenzteils ignorieren.
- Achten Sie darauf, dass Ihre Abgabe grundsätzlich ausführbar ist. Nicht ausführbare Abgaben werden mit 0 Punkten bewertet.

In dieser Aufgabe sind Sie an der Entwicklung eines Warenwirtschaftssystems für ein Versandhaus für Schreibwaren beteiligt. Dabei werden Sie insbesondere den Umgang mit den Datenstrukturen aus dem `Collection`-Framework der Java Base Class Library vertiefen. Achten Sie darauf, in dieser Aufgabe nur Datenstruktur aus `Collection`-Framework zu verwenden.

Ein Versandhaus (`Company`) nimmt neue Schreibwaren (`StationeryItem`) als Einzelstücke entgegen und lagert die Einzelstücke in einem Lagerregal (`StorageRack`) ein. Wenn eine Bestellung aufgegeben wird, wird das bestellte Einzelstück aus dem Lagerregal des Versandhauses entnommen, verarbeitet und kann danach verpackt und verschickt werden. Es gibt drei Arten von Schreibwaren: Kugelschreiber, Lineale und Zirkel. Diese werden durch die Kindklassen `Pen`, `Ruler` und `Compass` modelliert. Jede Instanz der eben genannten Klassen ist ein Einzelstück im Versandhaus.

In den folgenden Teilaufgabe werden Sie die Klassen im Paket `homework.warehouse` bearbeiten und dadurch das Versandhaus Schritt für Schritt vervollständigen.

Hinweis: Denken Sie in allen Teilaufgaben an Dokumentation, Vor- und Nachbedingungen, das Überprüfen dieser mittels offensiver und defensiver Programmierung und an Schleifeninvarianten und -varianten.

Hinweis: In der gesamten Aufgabe dürfen und sollen Sie sinnvolle Hilfsoperationen hinzufügen, um die geforderten Funktionalitäten zu realisieren.

Hinweis: In dieser Aufgabe müssen Sie an mehreren Stellen Ihre Design Entscheidungen begründen. Geben Sie die Begründungen in der PDF ab.

Hinweis: Im Paket `homework.warehouse` finden Sie außerdem die Klasse `Main`, die ein Beispiel für die Interaktion mit einem Versandhaus zeigt. Nutzen Sie diese Klasse, um die Funktionalitäten Ihres Versandhauses zu überprüfen. Sie dürfen die Klasse `Main` nach Belieben verändern. Die Inhalte der Klasse werden nicht bewertet.

(a) *(4 Punkte)* Einzelstücke werden in einem Lagerregal gelagert. Ein Lagerregal hat eine feste Anzahl an Fächern (die Kapazität des Regals), die durch das Attribut `capacity` repräsentiert wird. In einem Lagerregal sind alle Fächer nummeriert und jedes Fach kann maximal ein Einzelstück enthalten.

   i. Wählen Sie eine geeignete Datenstruktur aus, um die nummerierten Regalfächer zu realisieren. Verwenden Sie dabei eine Datenstruktur *ohne* Schlüssel-Wert-Paare. Begründen Sie Ihre Antwort kurz.

   ii. Fügen Sie der Klasse `StorageRack` eine solche Datenstruktur als Attribut hinzu und initialisieren Sie die Datenstruktur im Konstruktor. Die Einträge der Datenstruktur sollen vom Typ `Optional<StationeryItem>` sein und nach der Initialisierung eines Lagerregals mit leeren `Optional`-Objekten gefüllt sein (`Optional.empty()`).

   Hinweis: Die Verwendung der Klasse `Optional` wird im Foliensatz 6 auf den Folien 59 ff. ("*Optionale (Objekt-)Referenzen seit Java 1.8*") beschrieben.

(b) Implementieren Sie die folgenden Operationen der Klasse `StorageRack`:

   i. *(5 Punkte)* `addItem(StationeryItem stationeryItem)`: Fügt das als Parameter übergebene Einzelstück der Datenstruktur des Lagers am ersten freien Lagerplatz (also dem freien

Platz mit der niedrigsten Platznummer) hinzu. Dabei gilt, dass ein Regalfach genau dann frei ist, wenn der entsprechende Eintrag in der Datenstruktur `Optional.empty()` ist
Hinweise: Beachten Sie, dass Sie der Datenstruktur nur Objekte vom Typ `Optional` hinzufügen können.

ii. *(3 Punkte)* `removeItem(int compartmentNumber)`: Entfernt das Einzelstück am gegebenen Lagerplatz aus der Datenstruktur des Lagers. Wenn das Einzelstück nicht vorhanden ist, passiert nichts.

iii. *(3 Punkte)* `getItem(int compartmentNumber)`: Gibt das Einzelstück am gegebenen Lagerplatz zurück.
Hinweis: Beachten Sie, dass ein `Optional`-Objekt zurückgegeben werden muss, um auch den Fall behandeln zu können, dass der Lagerplatz frei ist.

(c) *(4 Punkte)* Jedes Einzelstück hat eine eindeutige Kennzeichnung (`Identifier`). Will man nun ein bestimmtes Einzelstück anhand der Kennzeichnung im Lager finden, so müsste man dafür das gesamte Lager durchsuchen.

Da das ineffizient ist, gibt es für jedes Lagerregal eine Datenstruktur, in der zu jeder Kennzeichnung hinterlegt ist, bei welcher Fachnummer das zugehörige Einzelstück zu finden ist. Damit kann man zu jeder Kennzeichnung schnell das zugehörige Lagerfach finden (und somit das Einzelstück selbst).

i. Wählen Sie eine geeignete Datenstruktur aus, um die Zuordnung von Kennzeichnung zu Fachnummer zu realisieren. Begründen Sie Ihre Antwort kurz.

ii. Fügen Sie der Klasse `StorageRack` eine solche Datenstruktur als Attribut hinzu und initialisieren Sie diese im Konstruktor.

(d) Erweitern Sie die folgenden Operationen der Klasse `StorageRack`:

i. *(1 Punkt)* Ergänzen Sie Ihre Operation `addItem(StationeryItem stationeryItem)` so, dass die Zuordnung von der Kennzeichnung des neuen Einzelstücks zu dessen Fachnummer in der in Teilaufgabe (c) ergänzten Datenstruktur gespeichert wird.

ii. *(1 Punkt)* Ergänzen Sie `removeItem(int compartmentNumber)` so, dass die Zuordnungen von der Kennzeichnung eis Einzelstücks zu dessen Fachnummer beim Herausnehmen eines Einzelstücks auch gelöscht wird.
Hinweis: Zum Löschen der Zuordnung benötigen Sie die Kennzeichnung des zu entfernenden Einzelstücks.

iii. *(3 Punkte)* Implementieren Sie außerdem die Abfrage `getCompartmentNumberOf(Identifier identifier)`, die die Fachnummer zurückgibt, in dem das im Einzelstück mit der als Parameter gegebenen Kennzeichnung liegt. Nutzen Sie hierfür die Datenstruktur aus Teilaufgabe (c). Mit dieser Abfrage kann nun - wie in vorherigen Teilaufgabe motiviert - der Lagerplatz jedes Einzelstücks effizient ermittelt werden.
Hinweis: Auch hier soll ein `Optional`-Objekt zurückgegeben werden, da das gegebene Identifikationsobjekt evtl. nicht im Lager zu finden ist.

(e) Die Klasse `Company` repräsentiert ein Versandhaus. Dort ist bereits das Attribut `itemStorageRack` für ein Lagerregal vorgegeben.

i. *(1 Punkt)* Implementieren Sie den Konstruktor der Klasse `Company`, sodass ein `StorageRack`-Objekt erzeugt und dem Attribut zugewiesenen wird. Gehen Sie davon aus, dass maximal 75 Einzelstücke gleichzeitig gelagert werden können.

ii. *(3 Punkte)* Implementieren Sie außerdem die Operation `storeInStorageRack(StationeryItem stationeryItem)` dieser Klasse, sodass das als Parameter übergebene Einzelstück direkt im Lager untergebracht wird. Falls ein Einzelstück nicht mehr ins Lager passt, wird es ignoriert.

(f) *(4 Punkte)* Wenn ein Einzelstück bestellt wird, wird es zunächst aus dem Lager genommen und in einen Pufferspeicher gelegt. Von dort werden die Einzelstücke nacheinander einer Verpackungsanlage zugeführt, sobald diese bereit ist. Diesem Pufferspeicher werden Einzelstücke von oben zugeführt und von unten entnommen.

i. Wählen Sie eine geeignete Datenstruktur aus, um dieses Verhalten möglichst einfach zu realisieren. Begründen Sie Ihre Antwort kurz.

ii. Fügen Sie der Klasse `Buffer` eine solche Datenstruktur als Attribut hinzu und initialisieren Sie die Datenstruktur im Konstruktor.

(g) Implementieren Sie in der Klasse `Buffer` die folgenden Operationen.

i. *(2 Punkte)* Die Abfrage `isEmpty()`, die zurückgibt, ob der Pufferspeicher leer ist oder nicht.

ii. *(2 Punkte)* Die Operation `bufferItem(StationeryItem stationeryItem)`, die ein Einzelstück dem Pufferspeicher hinzufügt.

iii. *(2 Punkte)* Die Operation `releaseItem()`, die dem Pufferspeicher das unterste Einzelstück entnimmt und dieses zurückgibt.

(h) *(6 Punkte)* Implementieren Sie in der Klasse `Company` die Operation `processIncomingOrder(Identifier identifier, Customer customer)`.

Die Operation entnimmt das Einzelstück mit der gegebenen Kennzeichnung aus seinem Fach im Lagerregal und fügt es dem Pufferspeicher hinzu. Der Pufferspeicher ist im Attribut `orderBuffer` gegeben. Falls sich das gegebene Einzelstück nicht im Lager befindet, wird die Bestellung ignoriert.

Hinweis: Ignorieren Sie in dieser Teilaufgabe vorerst den Parameter `customer`.

(i) *(6 Punkte)* Erweitern Sie die Funktionalität der Operation `processIncomingOrder(Identifier identifier, Customer customer)` so, dass alle Neukunden ein Werbegeschenk bekommen.

Dazu muss sich das Versandhaus merken, welche Kunden schon einmal etwas bestellt haben.

i. Wählen Sie eine geeignete Datenstruktur aus, um die bereits bekannten Kunden zu speichern. Begründen Sie Ihre Antwort kurz.

ii. Fügen Sie zur Klasse `Company` eine geeignete Datenstruktur als Attribut hinzu und initialisieren Sie diese im Konstruktor.

iii. Nutzen Sie die neue Datenstruktur, um zu überprüfen, ob der übergebene Kunde bereits bekannt, oder ein Neukunde ist. Falls letzteres zutrifft, bekommt der Kunde ein Werbegeschenk. Nutzen Sie die Operation `getBonusItem()` um ein Werbegeschenk zu generieren und fügen Sie das generierte Geschenk direkt nach dem eigentlich bestellten Einzelstück in den Pufferspeicher ein.
Hinweis: Denken Sie auch daran, die Datenstruktur mit den Kunden zu updaten.

3 Anhang - Maven Projekte erstellen und exportieren

Die folgenden Anweisungen brauchen Sie nicht als Vorbereitung für den Teil A, sondern, um die Lösungen des Teil B diese Blattes korrekt abzugeben. Weiterhin umfassen die folgenden Anweisungen der Vollständigkeit halber mehr als die für Teil B benötigten Kenntnisse.

3.1 IntelliJ

3.1.1 Maven Projekt in IntelliJ erstellen

Um in IntelliJ ein einfaches Maven Projekt zu erstellen, befolgen Sie die folgenden Schritte:

1. Klicken Sie auf File → New → Project
2. Wählen Sie in der linken Spalte Java aus.
3. Im rechten Bereich können Sie dem Project einen Namen geben und einen Speicherplatz wählen. Bei `Build system` wählen Sie `Maven` aus.
4. Unter `JDK` müssen Sie nun eine JDK auswählen, welches für das Projekt verwendet werden soll. Standardmäßig hatten wir bisher Version 21 genutzt, das können Sie zu Vereinfachung so belassen.
5. Unter `Advanced Settings` können Sie Ihrer `GroupId` und `ArtifactId` einen individuellen Wert geben, können es aber auch so belassen.
6. Mit Create erstellen Sie Ihr einfaches Maven Projekt.

3.1.2 Maven Projekt unter IntelliJ exportieren

1. Öffnen Sie das Projekt im Dateimanager. Gehen Sie dazu wie folgt vor: Rechtsklick auf den Projektordner (hier: `exercise-sheet-10`) im Projekt Explorer → Open In → Reveal in File Explorer beziehungsweise → Reveal in Finder bei MacOS.
2. Sie sollten nun einen Ordner mit dem Namen `exercise-sheet-10` in Ihrem Dateimanager sehen.
3. Verpacken Sie diesen Ordner als .zip.
4. *(Optional)* Öffnen Sie das erzeugte .zip Archiv und überprüfen Sie, dass alle Dateien aus dem Projekt enthalten sind.

3.2 Eclipse

3.2.1 Maven Projekt in Eclipse erstellen

Um in Eclipse ein einfaches Maven Projeckt zu erstellen, befolgen Sie die folgenden Schritte:

1. Gehen Sie auf File → New → Project
   Dort öffnet sich eine Listenansicht, in der Sie den Ordner `Maven` sehen können. Öffnen Sie diesen.
2. In diesem Ordner sehen Sie eine Auwahl an Möglichkeiten. Wir benötigen Maven Project.
3. Im nächsten Fenster können Sie entscheiden, wo das Projekt gespeichert werden soll. Im Normalfall können Sie den Haken bei `Use default Workspace Location` gesetzt lassen.
4. Zudem setzen Sie einen Haken bei `Create a Simple Project`.
   Klicken Sie auf Next
5. In diesem Feld können Sie die wichtigsten Namen vergeben. Sie müssen die Felder `GroupId`, `ArtifactId` und `Name` ausfüllen. Die `GroupId` und `ArtifactId` sind in der Regel wie die Namen von packages aufgebaut (umgekehrter Domänenname). Sie können sich für Ihr Projekt individuelle aber eindeutige Namen und IDs ausdenken.
6. Haben Sie dies erledigt, kann mit Finish ein einfaches Maven Projekt erstellt werden.

3.2.2 Maven Projekt unter Eclipse exportieren

1. **Exportieren Sie ein Projekt als .zip** mit File → Export, wählen Sie General → Archive File und drücken Sie auf Next. Wählen Sie Ihr Project in der Liste links oben aus und stellen Sie sicher, dass ein blauer Haken vor allen Dateien und Packages ist, indem Sie das Projekt ausklappen. Geben Sie im Textfeld To archive file einen Pfad und Namen an (Pfad\name.zip), wählen Sie unter Options Save in zip format und drücken Sie Finish.

2. *(Optional)* Öffnen Sie das erzeugte .zip Archiv und überprüfen Sie, dass alle Dateien aus dem Projekt enthalten sind.

3.3 VSCode

3.3.1 Maven Projekt in VSCode erstellen

Um in VSCode ein einfaches Maven Projekt zu erstellen, befolgen Sie die folgenden Schritte:

1. Stellen Sie sicher, dass das `Extension Pack for Java` installiert ist.

2. In dem Reiter Explorer klicken Sie Create Maven Project.

3. Nun wählen Sie `maven-archetype-quickstart` und dann `1.4`.

4. Nun können Sie in `GroupId` und `ArtifactId` dem Projekt einen Namen geben, und dann im Folgendem Fenster einen Ordner auswählen in dem das Projekt erstellt wird.

5. Nun gibt es im **Terminal** noch zwei weitere Eingabeprompts in denen Sie im Ersten eine `Versionsnummer` vergeben und im Zweiten Ihre gewählten Einstellungen bestätigen.

3.3.2 Maven Projekt unter VSCode exportieren

1. Öffnen Sie das Projekt im Dateimanager. Gehen Sie dazu wie folgt vor: Shift+Alt+R oder Rechtsklick auf den Projektordner (hier: `exercise-sheet-10`) im Projekt Explorer → Reveal in File Explorer beziehungsweise → Reveal in Finder bei MacOS.

2. Sie sollten nun einen Ordner mit dem Namen `exercise-sheet-10` in Ihrem Dateimanager sehen.

3. Verpacken Sie diesen Ordner als .zip.

4. *(Optional)* Öffnen Sie das erzeugte .zip Archiv und überprüfen Sie, dass alle Dateien aus dem Projekt enthalten sind.