

Funktionale und objektorientierte Programmierkonzepte

Übungsblatt 11



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Entwurf

Achtung: Dieses Dokument ist ein Entwurf und ist noch nicht zur Bearbeitung/Abgabe freigegeben. Es kann zu Änderungen kommen, die für die Abgabe relevant sind. Es ist möglich, dass sich **alle** Aufgaben noch grundlegend ändern. Es gibt keine Garantie, dass die Aufgaben auch in der endgültigen Version überhaupt noch vorkommen und es wird keine Rücksicht auf bereits abgegebene Lösungen genommen, die nicht die Vorgaben der endgültigen Version erfüllen.

Hausübung 11

Running a Business

Gesamt: 16 Punkte

Beachten Sie die Seite *Verbindliche Anforderungen für alle Abgaben im Moodle-Kurs*.

Verstöße gegen verbindliche Anforderungen führen zu Punktabzügen und können die korrekte Bewertung Ihrer Abgabe beeinflussen. Sofern vorhanden, müssen die in der Vorlage mit TODO markierten crash-Aufrufe entfernt werden. Andernfalls wird die jeweilige Aufgabe nicht bewertet.

Die für diese Hausübung relevanten Verzeichnisse sind `src/main/java/h11` und ggf. `src/test/java/h11`.

Einleitung

Die Vorlage besteht aus den folgenden Klassen:

- Employee
- Department
- Product
- Warehouse
- Company:
Die Klasse besitzt ein `private`-Objektattribut `departments` vom Typ `List<Department>`, sowie ein `private`-Objektattribut `warehouses` vom Typ `List<Warehouse>`.

H1: Das Department**?? Punkte**

H1.1: Title**?? Punkte**

Implementieren Sie in dieser Aufgabe die **public**-Objektmethode `getListOfPositionsInDepartment`. Diese Methode hat keinen Parameter und liefert ein Objekt vom Typ `List<Position>` zurück. Die Methode soll eine Liste aller Positionen, welche im Department enthalten sind, zurückliefern. Achten Sie dabei darauf, dass keine Position doppelt enthalten ist.

Hinweis:

Gucken Sie sich hier nochmal in der Dokumentation von Stream die folgende Methode an:

- `java.util.stream#distinct`

H1.2: Title**?? Punkte**

Implementieren Sie nun die **public**-Objektmethode `filterEmployeeByPosition`, welche einen formalen Parameter `position` vom Typ `Position` besitzt und als Rückgabety `List<Employee>` hat. Die Rückgabe der Methode soll eine Liste aller Angestellten sein, welche die im aktuellen Parameter übergebenen Position besitzen.

H1.3: Title**?? Punkte**

Implementieren Sie die **public**-Objektmethode `getNumberOfEmployeesBySalary`, welche einen formalen Parameter `salary` vom Typ `double` besitzt und als Rückgabety `long` hat. Die Methode liefert einfach die Anzahl aller Angestellten zurück, welche ein Gehalt größer oder gleich dem im aktuellen Parameter gegebenen Wert haben.

H1.4: Title**?? Punkte**

Implementieren Sie die **public**-Objektmethode `adjustSalary`, welche einen formalen Parameter `amount` vom Typ `double` und einen zweiten formalen Parameter `increase` vom Typ `boolean` hat und nichts zurückliefert. Die Methode soll für jeden Angestellten in dem Department das entsprechende Gehalt, um die im ersten aktuellen Parameter angegebene Menge erhöhen oder verringern, je nach Wert im zweiten aktuellen Parameter.

H2: Das Warenhaus**?? Punkte**

H2.1: Title**?? Punkte**

Implementieren Sie dazu die **public**-Objektmethode `getPrice`. Diese besitzt einen formalen Parameter `product` vom Typ `Product` und liefert `double` zurück. Die Rückgabe der Methode soll einfach der Preis des im aktuellen Parameter übergebenen Produktes sein. Achten Sie dabei darauf, dass, wenn der übergebene Parameter `null` ist, Sie dann einen Preis von 0.0 zurückliefern.

Verbindliche Anforderung:

In dieser Aufgabe dürfen Sie keine `if-else`-Statements verwenden. Sie müssen mit `Optinal` arbeiten.

Hinweis:

Schauen Sie sich für diese Aufgabe in `Optinal`-Dokumentation die Methoden `ofNullable` und `orElse` an.

H2.2: Title**?? Punkte**

Implementieren Sie die `public`-Objektmethode `getTotalQuantityOfProduct`. Die Methode besitzt den Rückgabetyt `long` und hat einen formalen Parameter `product` vom Typ `Product`. Die Methode soll die Gesamtmenge des im formalen Parameter übergebenen Produktes in dem aktuellen `Warehouse`-Objekt zurückliefern.

H2.3: Title**?? Punkte**

In dieser Aufgabe implementieren Sie die `public`-Objektmethode `getTotalPrice`, welche als Rückgabetyt `double` hat und keine formalen Parameter besitzt. Die Methode soll die Summe der Preise aller Produkte im aktuellen `Warehouse`-Objekt zurückliefern.

H2.4: Title**?? Punkte**

Implementieren Sie nun die `public`-Objektmethode `generateProducts`. Die Methode besitzt drei formalen Parameter. Der erste `type` vom Typ `ProductType`, der zweite `price` vom Typ `double` und der dritte `name` vom Typ `String`. Der Rückgabetyt der Methode ist `Stream<Product>`. Die Methode soll einen `Stream` erzeugen, welcher beliebig viele Objekte vom Typ `Product`, mit den in den aktuellen Parametern übergebenen Spezifikationen, erzeugt.

Hinweis:

Gucken Sie sich hier nochmal in der Dokumentation von `Stream` die folgende Methode an:

- `java.util.stream#generate()`

H2.5: Title**?? Punkte**

Implementieren Sie jetzt die `public`-Objektmethode `addProducts`, welche einen formalen Parameter `product` vom Typ `Product` und einen formalen Parameter `numberOfProducts` vom Typ `int` besitzt. Die Methode soll nun Produkte aus dem, in der vorherigen Aufgabe implementierten, `Stream<Product>` zu dem Attribut `products` hinzufügen, und zwar so viele Elemente, wie der Wert im aktuellen Parameter `numberOfProducts` vorgibt.

Hinweis:

Gucken Sie sich hier nochmal in der Dokumentation von `Stream` die folgende Methode an:

- `java.util.stream#limit()`

H3: Die Firma**?? Punkte**

H3.1: Title**?? Punkte**

In dieser Aufgabe implementieren Sie die **public**-Objektmethode `getListOfAllEmployee`, welche keine Parameter besitzt und den Rückgabetyt `List<Employee>` hat.

Die Methode soll eine Liste aller Angestellten aus allen Departments zurückliefern.

Hinweis:

Gucken Sie sich hier nochmal in der Dokumentation von Stream die folgende Methode an:

- `java.util.stream#flatMap(java.util.function.Function)`

H3.2: Title**?? Punkte**

Implementieren Sie die **public**-Objektmethode `getQuantityOfProduct`, welche als Rückgabetyt **long** hat und einen formalen Parameter `product` vom Typ `Product` besitzt.

Die Methode soll die gesamte Anzahl des im aktuellen Parameter übergebene Produktes aus allen Warenhäusern zurückliefern.

H3.3: Title**?? Punkte**

AUFGABE ZUM ZUSAMMENFÜGEN (maybe)

H3.4: Title**?? Punkte**

Implementieren Sie die **public**-Objektmethode `getFilteredProductNames`. Die Methode hat den Rückgabetyt `List<String>` und besitzt einen formalen Parameter `predicates` vom Typ `List<Predicate<Product>>`.

Die Rückgabe der Methode soll eine Liste mit allen Produktnamen sein, aus allen Warenhäusern, welche alle Prädikate aus dem aktuellen Parameter erfüllen.

H3.5: Title**?? Punkte**

Implementieren Sie die **public**-Objektmethode `productsInPriceRange`. Die Methode besitzt zwei formale Parameter `low` und `high` beide vom Typ **double**. Der Rückgabetyt der Methode ist `List<Product>`.

Die Rückgabe der Methode soll eine Liste aller Produkte aus allen Warenhäusern sein, deren Preis in dem Intervall `[low, high]` liegt. Außerdem soll die Liste aufsteigend sortiert sein.

H3.6: Title**?? Punkte**

Implementieren Sie nun die **public**-Objektmethode `getEmployeesSortedByName`, welche keine Parameter und als Rückgabetyt `List<String>` besitzt.

Die Methode soll eine Liste der Namen aller Angestellten aus allen Departments zurückliefern, welche aufsteigend

nach Nachnamen sortiert ist.

Außerdem sollen die `String`-Objekte wie folgt formatiert werden:

"Max Mustermann" → "Mustermann, Max"

Hinweis:

In dieser Aufgabe gibt es mehrer Möglichkeiten, wie Sie die Formatierung der Strings realisieren können. Sie können einmal auf `String` arbeiten, gucken Sie sich dafür in der Klasse `String` die `split`-Methode an. Sie können aber auch, wenn Sie wollen, mit regulären Ausdrücken arbeiten. Gucken Sie sich dafür einmal die Klasse `java.util.regex.Pattern` an.

H3.7: Title**?? Punkte**

Implementieren Sie nun abschließend die `public`-Objektmethode `getAllProductsByType`. Die Methode besitzt einen formalen Parameter `type` vom Typ `ProductType` und hat den Rückgabebetyp `List<String>`. Die Rückgabe der Methode soll eine Liste aller Produkte, des im aktuellen Parameter übergebenen Produkttypens, und dem entsprechenden Preis sein. Die Liste soll zudem absteigend sortiert sein und es sollen nur die im aktuellen Parameter `numberOfProducts` gegebene Anzahl an Produkten angezeigt werden.

Beispiel:

Ein String in der Rückgabe sollte dann wie folgt aussehen:

"Laptop: 1000€"