HSLU Hochschule Luzern

Objektorientierte Programmierung (OOP)

Übung SW13: Lambdas und Stream API

Themen: Funktionale Programmierung (O14), Stream API (O15).
Zeitbedarf: ca. 180min.

Roland Gisler, Version 1.5.0 (FS25)

1 Einsatz von Lambda-Expressions

1.1 Lernziele

- Refactoringvorschläge der Entwicklungsumgebung nachvollziehen können.
- Formulierung einfacher Lambda-Expressions.
- Verwendung von Methodenreferenzen in Lambdas.

1.2 Grundlagen

Als Grundlage dienen die Codebespiele aus dem Input `O14` und verschiedene Übungen aus den vergangenen Semesterwochen. Es handelt sich dabei primär um Übungen, in welchen wir potentiell oder tatsächlich [anonyme] innere Klassen verwenden konnten:

- O09: Implementation von **`Comparable`**, Einsatz von Collections (Sortierung).
- E02/E03: Unit-Testing, speziell Testing von Exceptionhandling mit JUnit 5.
- O12: Eventhandling und innere Klassen.

Dieser Übungsblock ist somit auch eine gute Gelegenheit für einen Rückblick und Repetition!

Zusätzliche empfehle ich Ihnen den folgenden, relativ kurzen Quickstart-Guide von Oracle:
http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/Lambda-QuickStart/index.html
(Hinweis: Bezieht sich noch auf Java 8, ist aber völlig ok als Einstieg.)

1.3 Aufgaben

a.) Ein guter Einstieg sind die Beispiele mit dem **`PropertyChangeListener`** aus den Übungen zum Eventhandling (O12), die Sie auch mit Hilfe des Inputs nachvollziehen können: Ersetzen Sie die inneren Klassen durch Lambda-Expressions. Gehen Sie dabei am Besten schrittweise vor, und vereinfachen Sie den Quellcode Stück für Stück.

b.) Beim Eventhandling sind Lambdas besonders praktisch. Oftmals lohnt es sich die Behandlung in eine kleine Methode auszulagern (was Sie idealerweise schon so gemacht haben). Dann kann das Lambda als einfache Methodenreferenz formuliert werden!

c.) In O09 haben wir Komparatoren implementiert, und diese in D02 dann zur Sortierung von Temperaturwerten verwendet. Hier ist das Potential besonders gross: Wir können die freistehende Klasse, welche das **`Comparator`**-Interface implementiert vollständig durch eine Lambda-Expression ersetzen. Nutzen Sie dafür Ihre **`Person`**-Klasse, erzeugen Sie ein paar Personen und legen Sie diese in eine Collection ab. Dann können Sie sortieren. Probieren Sie es aus!

d.) Experimentieren Sie frei mit verschiedenen Implementationen. Lambdas machen (wie anonyme innere Klassen) speziell dann Sinn, wenn wir sie **nur einmal** ad-hoc brauchen. Aber Achtung: Sie können Lambdas auch Variablen zuweisen und speichern (siehe Input!)

2 Stream API – Tutorial und Übung «Temperaturverlauf»

2.1 Lernziele

- Verwendung der Stream API der Collection-Klassen.
- Formulierung einfacher Lambda-Expressions.
- Funktionale, deklarative Programmierung.

2.2 Grundlagen

Als Grundlage dienen die Codebespiele aus dem Input `O15` und das folgende Tutorial von Benjamin Winterberg: https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/

Beachten Sie auch das Kapitel 5 «Functional Processing of Collections (Advanced)» in unserem Lehrbuch «Objects first with Java»!

2.3 Aufgaben

a.) Studieren Sie das oben angegebene, sehr gute Tutorial. Vollziehen Sie so viele Beispiele wie möglich nach, indem Sie sie selbst ausprobieren. Es lohnt sich! Experimentieren Sie!

b.) Nehmen Sie sich die Zeit und studieren Sie die originale JavaDoc zumindest von ausgewählten Interfaces bzw. Klassen wie z.B.
`Stream:` https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/stream/Stream.html

`IntStream:`
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/stream/IntStream.html
Schauen Sie sich auch die Functional Interfaces an, die als Methodenparameter verwendet werden, zum Beispiel
`Predicat:`
https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/function/Predicate.html

c.) Nehmen Sie die Übung zu O08 (**`TemperaturVerlauf`**) zur Hand und versuchen Sie diese alternativ mit Hilfe der Stream API zu refaktorieren. Ein erster, einfacher Einstieg sind die Iterationen für die Ausgabe: Holen Sie mit **`stream()`** auf der **`Collection`** den «Elemente-Strom» und verwenden Sie die Methode **`forEach()`** (und eine Lambda-Expressions) um verschiedene Ausgaben zu produzieren.

d.) Stellen Sie sich eigene Aufgaben, zum Beispiel:
- Filtern Sie bestimmte Werte aus.
- Suchen Sie Maxima und Minima.
- Konvertieren Sie in andere Objekt(-typen).
- Zählen Sie wie oft ein bestimmter Wert (oder Wertebereich) enthalten ist.

  etc.

Empfehlung: Für ein maximales Verständnis implementieren Sie die Lösung zuerst immer «klassisch» (also imperativ) und vergleichen sie dann mit der auf der Stream API und Lambdas basierenden Lösung. Vorhandene (oder spätestens jetzt aber noch vor dem Refactoring ergänzte) Testfälle geben Ihnen die notwendige Rückendeckung!

3 Optionale Repetitionsaufgabe: Raumverwaltung – Teil 4 von 5

3.1 Lernziele

- Repetition bereit behandelter Themen zur Festigung und Vertiefung.

3.2 Grundlagen

Diese Zusatzaufgaben sind unabhängig von den restlichen Aufgaben dieser Woche. Sie baut aber auf der optionalen Repetitionsaufgabe der letzten Woche auf.

Optionale Aufgaben sind **nicht** Bestandteil der Besprechungen.

3.3 Aufgaben

a.) Ergänzen Sie im Konstruktor der Klasse **`RaumVerwaltung`** die folgenden Räume als Testdaten:

Raumnummer	Kapazität	Hinweis
600	18	
602	6	
603	12	Schon vorhanden aus einer früheren Aufgabe
605	24	
610	12	

b.) Erstellen Sie eine Klasse **`Demo`** mit einer **`main(…)`**-Methode. Instanzieren Sie in dieser ein Objekt **`RaumVerwaltung`** und reservieren Sie je einen Raum für **`11`**, **`6`** und **`17`** Personen. Geben Sie danach ein Liste aller Räume auf der Konsole aus, und prüfen Sie, ob nun drei Räume belegt sind.

c.) Optimieren Sie die Implementation der Methoden auf der Klasse **`RaumVerwaltung`**. Überarbeiten Sie die Reservation derart, dass für eine Anforderung der jeweils kleinste noch passende Raum gewählt wird.

d.) Überladen Sie auf der Klasse **`Raum`** (und ggf. weiteren Klassen) die notwendigen Methoden, damit ein **`Raum`** einfach mit **`System.out.println(raum)`** in einem für Logging geeigneten Format ausgegeben werden kann.

e.) Überladen Sie auf der Klasse **`RaumVerwaltung`** die Methode zur Freigabe einer Reservation so, dass damit ein Raum auch nur anhand seiner Raumnummer freigegeben werden kann.