Übungsblatt 3: Mehr Collections und Generics

Abgabetermin: 19.5.2017, 23.59 Uhr

Aufgabe 1: HashMap vs Array

(4 + 1 + 1)

`HashMaps` [2] sind eine Datenstruktur die es ermöglicht einem Schlüssel einen Wert zu zu weisen. In dieser Hinsicht sind sie eine Verallgemeinerung von Arrays indem sie nicht nur natürliche Zahlen als Schlüssel zulassen.

1. Zeige dass HashMaps als Verallgemeinerung von Arrays gesehen werden können, indem du eine Klasse `HashMapArray` definierst, die sich wie ein Array verhält und die typischen Array Operationen (hinzufügen an einem Index, Element an einem Index abfragen, Länge abfragen, BoundsExceptions) aufweist aber intern über eine `HashMap` arbeitet.
2. Es gibt Gründe warum in Java Arrays nicht als HashMap implementiert sind. Einer davon ist die historische Entwicklung. Welche anderen Nachteile hat `HashMapArray` im Vergleich zu einem 'normalen' Array?
3. Natürlich gibt es Fälle bei denen die Verwendung einer `HashMap` sinnvoller ist als die eines Array. Was sind diese Gründe?

Aufgabe 2: HashMap in Action (1 + 1 + 2 + 3)

In dieser Aufgabe widmen wir uns einer konkreten Verwendungsmöglichkeit von `HashMap`: Eine Notenübersicht.

Konkret wollen wir eine Möglichkeit schaffen Noten und erworbene LPs ihren zugehörigen Fächern zuzuteilen. Dafür bietet sich der folgende Typ an: `HashMap<String, Pair<Double,Int>>`.

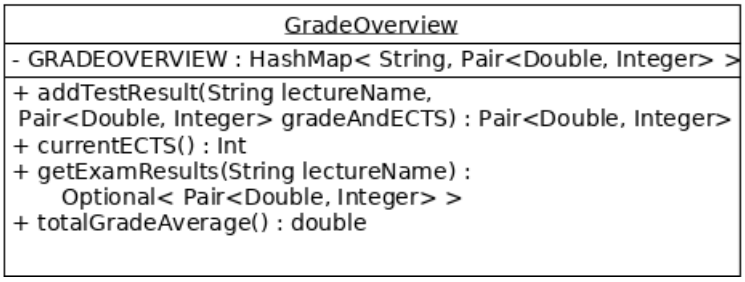

Abbildung 1: Klassendiagramm für GradeOverview

1. Erstelle die Klasse `GradeOverview` mit dem gleichnamigen Feld und der Methode `addTestResult`.
2. Implementiere die Methode `currentECTS` welche die aktuell akkumulierten LPs zurück gibt.
3. Implementiere die Methode `getExamResults` welche die Note und erworbenen LPs zu einer Prüfung zurück gibt insofern diese Vorhanden ist. Informiere dich hierzu über das `Optional<T>` Konstrukt [1].
4. Implementiere die Methode `totalGradeAverage` welche die Gesamtnote berechnet. Beachte dabei, dass die Noten anhand der LPs gewichtet sind.
5. Teste deine Implementierung mit sinnvollen Werten.

Aufgabe 3: Bonus: Streams und Funktionen (2)

Die *Java Stream API* [3] erlaubt 'die Verarbeitung von Elementströmen in einem funktionalen Stil'. Streams bieten im Vergleich zu dem altbekannten 'for-Loop' Stil einige Vorteile wie Expressivität oder nativen Support für Parallelität.

1. Informiere dich über Streams und verwende sie um die Methoden der Klasse `GradeOverview` zu implementieren bei denen die Verwendung Sinn ergibt. Tipp: Lambdas sind in diesen Kontext hilfreich [4].

Abgabe

Beachte bei der Abgabe folgendes:

- Die Bearbeitung kann in Gruppen erfolgen, muss jedoch nicht. Wichtig ist, dass, sollte die Abgabe in einer Gruppe erfolgen die Namen der Mitglieder erkenntlich sein sollten.
- Die Abgabe erfolgt über Moodle.
- Der abgegebene Code muss kompilieren.
- Alle von Ihnen erzeugten Klassen und Methoden müssen ausreichend mit JavaDoc und normalen Kommentaren versehen sein.
- Alle Fragen müssen hierbei sinnvoll beantwortet sein.

Fragen

Wenn du Fragen hast, nutze bitte das Forum im Moodle Kurs oder frage deinen Tutor.

Literatur

[1] https://docs.oracle.com/javase/8/docs/api/java/util/Optional.html

[2] https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

[3] https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html

[4] https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html