Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Programmieren

Prof. Dr. Larissa Putzar &
Thorben Ortmann

Collections Aufgaben

Vorwort

- **Schreiben Sie zu jeder der zu implementierenden Aufgaben/Methoden Testfälle.**
- Gehen Sie nach der TDD-Maxime vor: Red-Green-Refactor.
- Schreiben Sie Ihre Testfälle im Stile des AAA-Pattern.
- **Schreiben Sie zu jeder der zu implementierenden Aufgaben/Methoden JavaDoc.**
- Es gibt Aufgaben für Listen, Mengen und Maps.
- Es wird empfohlen 3 Klassen (ListUtils, SetUtils und MapUtils) anzulegen und die Aufgaben 1-3 über statische Methoden dieser Klassen zu lösen.
- Streams sind nicht explizit in den Aufgaben enthalten, aber für die Lösung einiger Aufgaben sehr hilfreich.

- Falls Sie Hilfe beim grundsätzlichen Aufsetzen Ihres Java-Projektes benötigen, können Sie dieses Template in Eclipse importieren und darauf aufsetzen:
  - https://artemis.mt.haw-hamburg.de/playground_tor/sose2022programmieren2/exercises/1-collections

Aufgabe 1

1. Implementieren Sie eine Methode *getFifthElement()*, die stets den fünften Integer einer Liste von Integern zurückgibt.

2. Implementieren Sie eine Methode *sortAscending()*, die eine Liste von Integern aufsteigend sortiert und zurückgibt.

3. Implementieren Sie eine Methode *getMostFrequentElement()*, die den Integer zurückgibt, der am häufigsten in einer Liste von Integern enthalten ist.

Aufgabe 2

1. Implementieren Sie eine Methode *getBiggestInteger()*, die den größten Integer aus einer Menge von Integern zurückgibt.

2. Implementieren Sie eine Methode *intersection()*, die die Schnittmenge von zwei Mengen von Integern zurückgibt.

3. Implementieren Sie eine Methode *cartesianProduct()*, die das kartesische Produkt zweier Mengen von Integern zurückgibt.

Aufgabe 3

1. Implementieren Sie eine Methode *getBiggestValue()*, die den größten Wert einer *Map<String,Integer>* zurückgibt.

2. Implementieren Sie eine Methode *sortKeysOfMapDescending()*, die die Menge der Schlüssel einer *Map<String,Integer>* in absteigender Reihenfolge zurückgibt.

3. Implementieren Sie eine Methode *mergeKeysAndValues()*, die die Vereinigungsmenge aus Schlüsseln und Werten einer *Map<Integer, Integer>* ausgibt.

Aufgabe 4

- Definieren Sie ein Interface *Buyable*. Das Interface soll die Operation *Integer fetchPrice()* vorgeben.
- Erstellen Sie eine Klasse *Car* und eine Klasse *Drink*, die jeweils das Interface *Buyable* implementieren.
- Im Konstruktor von *Car* bzw *Drink* soll der Preis gesetzt werden.
- *Car* und *Drink* sollen zusätzlich das Interface *Comparable* implementieren, damit *Buyable*-Objekte nach ihrem Preis sortierbar sind.
- Erstellen in einem Testfall eine *ArrayList* von *Buyable*-Objekten und fügen Sie 2 Getränke und 3 Autos zu der Liste hinzu (*Arrange*).
- Sortieren Sie die Liste aufsteigend (*Act*).
- Prüfen Sie, ob die List, wie erwartet sortiert wurde (*Assert*).

Aufgabe 5

Szenario: Sie wollen einkaufen gehen. Der Supermarkt Ihres Vertrauens stellt seine Waren in alphabetischer Reihenfolge aus. In Ihrer Einkaufsliste erfassen Sie den Namen der Artikel, die Sie kaufen möchten und die benötigte Anzahl.

Implementieren Sie eine Klasse *ShoppingList.* Diese besitzt nur ein Attribut *items* vom Typ *Map*.

- Überlegen Sie was Schlüssel und was Werte der Map sind.
- Fügen Sie Ihrer *ShoppingList* eine Methode *sort()* hinzu, die die Element in *items* als alphabetisch geordnete Liste von Einträgen (*Entry*)zurückgibt, sodass sie effizienter einkaufen können.