Java Intro Workshop

5 Aufgaben zu Collections

Aufgabe 1: Tennisverein-Mitgliederliste 10 min

Nehmen wir an, wir verwalten die Mitglieder des Tennisvereins in Form einer Liste. Implementieren Sie folgenden Registrierungsprozess: Zunächst melden sich Michael, Tim und Werner an. Prüfen Sie, ob sich Jana schon angemeldet hat. Nun registriert sich Andreas. Schließlich melden sich Lili, Jana und Natalija an. Geben Sie die Anzahl der Mitglieder aus.

Aufgabe 2: Check Magic Triangle 30 min

Schreiben Sie eine Methode `boolean isMagicTriangle(List<Integer> values)`, die prüft, ob eine Folge von Zahlen ein magisches Dreieck bildet. Ein solches ist definiert als ein Dreieck, bei dem die jeweiligen Summen der Werte der drei Seiten alle gleich sein müssen. Beginnen Sie mit einer Vereinfachung für die Seitenlänge 3 und einer Methode `boolean isMagic6(List<Integer> values)`.

Beispiele Nachfolgend ist das für je ein Dreieck der Seitenlänge drei und der Seitenlänge vier gezeigt:

```
  1          2
 6 5        8 5
2 4 3      4   9
          3 7 6 1
```

Damit ergeben sich folgende Seiten und Summen:

Eingabe	Werte 1	Werte 2
Seite 1	1 + 5 + 3 = 9	2 + 5 + 9 + 1 = 17
Seite 2	3 + 4 + 2 = 9	1 + 6 + 7 + 3 = 17
Seite 3	2 + 6 + 1 = 9	3 + 4 + 8 + 2 = 17

Tipp: Extrahieren Sie die Seiten mithilfe von `subList(startIdx, endIdx)`.

Aufgabe 3: Mengen und ihre Operationen 20 min

Zunächst ist es Ihre Aufgabe, Duplikate aus einer Liste mit Städtenamen zu entfernen:

```
List<String> cities = List.of("Kiel", "Hamburg", "Zürich", "Bremen",
                              "Hamburg", "Zürich", "Kiel", "Bremen");
```

Nun sollen Sie für folgende Zahlen die Mengenoperationen berechnen:

```
{ 1, 1, 2, 3, 5, 8, 13, 21 }
{ 1, 2, 3, 4, 5, 6, 7}
```

BONUS: Schreiben Sie eine generische Variante der symmetrischen Differenz, die wir auf den Folien kennengelernt haben. Starten Sie mit folgenden Zeilen:

```
static <T> Set<T> symDiff(Set<T> values1, Set<T> values2)
{
    // Schnittmenge bilden
    Set<T> intersection = // TODO

    // Vereinigung bilden und Schnittmenge abziehen
    Set<T> result = new TreeSet<>();
    // TODO
    return result;
}
```

```
System.out.println("Symm. Diff: " + symDiff(Set.of("A", "B", "C",
                                                  "D", "E", "F"),
                                       Set.of("D", "E", "F",
                                                  "G", "H", "I")));
```

```
=>

Symm. Diff: [A, B, C, G, H, I]
```

Aufgabe 4: Doppelte Buchstaben entfernen 20 min

Schreiben Sie eine Methode `String remove_duplicates(String input)`, die in einem gegebenen Text jeden Buchstaben nur einmal behält, also alle späteren doppelten unabhängig von Groß- und Kleinschreibung löscht. Dabei soll aber die ursprüngliche Reihenfolge der Buchstaben beibehalten werden.

Eingabe	Resultat
"bananas"	"bans"
"lalalamama"	"lam"
"MICHAEL	"MICHAEL"

Aufgabe 5: Duplikate entfernen 15 min

Aus einer Liste sollen Sie die doppelten Einträge entfernen. Dabei gilt die Randbedingung, dass die ursprüngliche Reihenfolge bestehen bleibt. Schreiben Sie dazu eine Methode `List<Integer> removeDuplicates(List<Integer> values)`.

Eingabe	Resultat
[1, 1, 2, 3, 4, 1, 2, 3]	[1, 2, 3, 4]
[7, 5, 3, 5, 1]	[7, 5, 3, 1]
[1, 1, 1, 1]	[1]

Tipp: Kombinieren Sie Listen und Mengen geeignet.

Aufgabe 6: Zahl als Text 20 min

Schreiben Sie eine Methode `String numberAsText(int n)`, die für eine gegebene positive Zahl vom Typ `int` die jeweiligen Ziffern in korrespondierenden Text umwandelt.

```
System.out.println(numberAsText(721971));
```

Das sollte folgende Ausgabe produzieren:

```
SEVEN TWO ONE NINE SEVEN ONE
```

Aufgabe 7: Morse Code 20 min

Schreiben Sie eine Methode `String toMorseCode(String input)`, die einen übergebenen Text in Morsezeichen übersetzen kann. Diese bestehen aus Sequenzen von ein bis vier kurzen und langen Tönen pro Buchstabe, symbolisiert durch ' .' oder ' -'. Zur leichteren Unterscheidbarkeit soll zwischen jedem Ton ein Leerzeichen und zwischen jeder Buchstabentonfolge jeweils drei Leerzeichen Abstand sein – ansonsten würden sich S (...) und EEE (...) nicht voneinander unterscheiden lassen. Beschränken Sie sich der Einfachheit halber auf die Buchstaben E, O, S, T, W mit folgender Codierung:

Buchstabe	Morsecode
E	.
O	- - -
S	. . .
T	-
W	. - -

Hier ein paar Beispiele für mögliche Umwandlungen:

Eingabe	Resultat
SOS	`...  ---  ...`
TWEET	`-  .--  .  .  -`
WEST	`.--  .  ...  -`

Aufgabe 8: Anagramm 15 min

Als Anagramm bezeichnet man zwei Strings, die dieselben Buchstaben in der gleichen Häufigkeit enthalten. Dabei soll Groß- und Kleinschreibung keinen Unterschied machen. Schreiben Sie eine Funktion `boolean isAnagram(String str1, String str2)`.

Eingabe 1	Eingabe 2	Resultat
"Otto"	"Toto"	True
"Mary"	"Army"	True
"Ananas"	"Bananas"	False

Aufgabe 9: Every2nd-Iterator 15 min

In dieser Aufgabe soll ein einfacher Iterator implementiert werden, der ausgehend von der Startposition, dann jeweils das 2. Element einer Datenstruktur ausgibt.

Für die Ausgangsdaten

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

erwarten wir folgendes Ergebnis: [1, 3, 5, 7, 9]

Aufgabe 10: EveryNth-Iterator 15 min

Implementieren Sie einen Iterator namens `EveryNth`, der jedes n-te Element durchläuft. Also für die Eingabe `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]` beispielsweise folgende Werte ausgibt:

- Mit Schrittweite 3: `1, 4, 7 ,10`
- Mit Schrittweite 5: `1, 6, 11`