Ue Lambda-Ausdrücke

Vorbereitung:

- Erstellen Sie ein neues Java-Projekt, erstellen Sie eine Main-Klasse.

Teilaufgabe 1

Erstellen Sie eine Methode `update()`, die eine Liste von Zahlen und eine Aktion (Function) als Parameter akzeptiert. Die Methode soll die Aktion in-place auf die Zahlenliste anwenden.

Signatur:

```
void update(List<Integer> numbers, Function<Integer, Integer> updater);
```

Beispielaufrufe in der main-Methode unter Anwendung von Lambda-Ausdrücken:

- Erhöhen Sie alle Werte um 1!
- Verdoppeln Sie alle Werte!
- Vermindern Sie alle Werte um 2!
- Ersetzen Sie die Werte durch den Abstand vom Mittelwert aller Werte!
- Geben Sie eine weitere Anwendung von update() an!

Teilaufgabe 2

Erstellen Sie eine Methode `filterNumbers()`, die eine Liste von Zahlen und eine Bedingung (Predicate) als Parameter akzeptiert. Die Methode soll alle Zahlen zurückgeben, die die Bedingungen erfüllen:

Signatur:

```
List<Integer> filterNumbers(List<Integer> numbers, Predicate<Integer> condition);
```

Beispielaufrufe in der main-Methode unter Anwendung von Lambda-Ausdrücken:

- Geben Sie alle geraden Zahlen aus!
- Geben Sie alle ungeraden Zahlen aus!
- Geben Sie alle Vielfachen von 4 aus!
- Geben Sie alle Primzahlen aus!
- Geben Sie eine weitere Anwendung von filterNumber() an!

Teilaufgabe 3

Erstellen Sie eine Methode `transformString()`, die eine Liste von Strings und eine Transformation (Function) als Parameter akzeptiert. Die Methode soll die transformierten Strings zurückgeben.

Signatur:

```
List<String> transformStrings(List<String> strings, Function<String, String>
transformer);
```

Beispielaufrufe in der main-Methode unter Anwendung von Lambda-Ausdrücken:

- Geben Sie die Strings in Großbuchstaben aus!
- Geben Sie die Strings in Kleinbuchstaben aus!
- Hängen Sie an jeden String „!“ an und geben Sie die Ergebnisse aus!
- Geben Sie die umgekehrten Strings aus!
- Geben Sie eine weitere Anwendung von transformStrings() an!

Teilaufgabe 4

Erstellen Sie eine Methode `processNumbers()`, die eine Liste von Zahlen, einen Filter und einen Transformer als Parameter akzeptiert. Die Methode für jene Elemente der Liste, die die Bedingung erfüllen, die Transformation anwenden und die Summe sich ergebenden Werte zurückgeben.

Signatur:

```
Integer processNumbers(List<Integer> numbers, Predicate<Integer> filter,
Function<Integer, Integer> transformer);
```

Beispielaufrufe in der main-Methode unter Anwendung von Lambda-Ausdrücken:

- Berechnen Sie die Summe aller Zahlen!
- Berechnen Sie die Summe der ungeraden Zahlen!
- Berechnen Sie die Summe des doppelten Werte aller ungeraden Zahlen!
- Geben Sie eine weitere Anwendung von processNumbers() an!

Teilaufgabe 5

Erstellen Sie eine Methode `groupByCondition()`, die eine Liste von Zahlen und eine Bedingung (Predicate) als Parameter akzeptiert. Die Methode soll eine Map zurückgeben, welche aus der gegebenen Zahlenliste zwei generiert: einer Liste, welche die Zahlen enthält, die die Bedingung erfüllen, und einer Liste mit jenen Zahlen, die die Bedingung nicht erfüllen

Signatur:

```
Map<Boolean, List<Integer>> groupByCondition(List<Integer> numbers, Predicate<In-
teger> condition);
```

Beispielaufrufe in der main-Methode unter Anwendung von Lambda-Ausdrücken:

- Differenzieren Sie die Zahlen nach gerade/ungerade.
- Differenzieren Sie die Zahlen nach Zahlen < 5 und Zahlen >= 5.
- Differenzieren Sie die Zahlen nach positiven und negativen Zahlen.
- Differenzieren Sie die Zahlen nach Primzahl/nicht Primzahl.
- Geben Sie eine weitere Anwendung von groupByCondition() an!