Übungen zu objektorientierte Programmierung 2

frieder.loch@ost.ch

Woche 2: Generics und Reflection

Aufgabe 1: Collection Hilfsfunktionen

Java bietet eine Vielzahl an Funktionen für Collections, dennoch fehlen einige nützliche Features. Es soll eine Reihe neuer **generischer Hilfsmethoden** für Collections implementiert werden.

Die nachfolgenden Funktionalitäten sollen als statische Methoden in der Klasse `CollectionFunctions` bereitgestellt werden. Die Methoden sind hier abgekürzt in Pseudocode beschrieben: Ihre Aufgabe ist es auch, eine geeignete generische Methodensignatur zu finden.

Methode	Beschreibung
`List<..> mergeToList(coll1, coll2)`, `Set<..> mergeToSet(coll1, coll2)`	Erzeugt eine neue Liste bzw. Menge bestehend aus den Elementen aus den Collections `coll1` und `coll2`.
`void merge(inputColl1, inputColl2, ..... targetColl)`	Fügt alle Elemente aus Collection `inputColl1` und `inputColl2` in die existierende Collection `targetColl` ein.
`Set<..> findDuplicates(coll)`	Findet alle Duplikate (Mehrfachvorkommen) in einer beliebigen Collection `coll`. **Beispiel:** { 1, 2, 1, 1, 3, 2, 4 } => Duplikate sind { 1, 2 }
`.. mostFrequent(coll)`	Bestimmt das/ein Element, welches am häufigsten mehrfach in der Collection `coll` enthalten ist. **Beispiel:** { 1, 2, 1, 1, 3, 2, 4 } => die 1 kommt am häufigsten vor

In der Vorlage finden Sie Testcode für Ihre Funktionen.

Fakultativ: - Benutzen Sie generische Varianz. Ansonsten können Sie den Testfall `testVariance()` weglassen.

Aufgabe 2: Varianzen

Erstellt eine Klasse `VarianceExamples`, die folgende Methoden enthält. Dabei sollt ihr Invarianz, Kovarianz (? extends T) und Kontravarianz (? super T) gezielt einsetzen. Die zu implementierenden Methoden sollen alle statisch sein.

Methode	Beschreibung
`double sum(Collection<...> numbers)`	Diese Methode berechnet die Summe aller Elemente in einer Collection von `Number` oder einer Unterklasse von `Number`.
`void addNumbers(List<...> list, List<...> source)`	Diese Methode befüllt eine `List`, die mindestens Integer-Werte aufnehmen kann (z. B. `List<Integer>`, `List<Number>`, `List<Object>`) aus der `source-List`

Methode	Beschreibung
`T findMax(Collection<T> coll)`	Findet das grösste Element in einer `Collection<T>`
`List<T> filterByType(Collection<?> source, Class<T> clazz)`	Diese Methode filtert eine `Collection<?>` und gibt eine neue `List<T>` mit allen Elementen des gewünschten Typs zurück. (**Tipp**: verwendet die Methode `clazz.isInstance(obj)` für die Laufzeittyp-Überprüfungen)