P. Thiemann, L. Fennell Sommersemester 2017

Programmieren in Java

`http://proglang.informatik.uni-freiburg.de/teaching/java/2017/`

food

Nahrungsmittel

Woche 06 Aufgabe 1/3

Herausgabe: 2017-05-29
Abgabe: 2017-06-17

Achtung: beachten Sie unbedingt die allgemeinen Hinweise zur Abgabe auf der Homepage.

Project `food`
Package `food`
Klassen

Food

`public Food (String name,`
`  double carbohydrateShare, double fatShare, double proteinShare)`
`public String getName()`
`public double relativeEnergyDensity()`
`@Override public boolean equals(Object r)`
`@Override public int hashCode()`

Meal

`public Meal(String name, Map<Food, double> ingredients)`
`public String getName()`
`public Map<Food, Double> getIngredients();`
`public double getCalorificValue();`

In dieser Aufgabe sollen Klassen für Nahrungsmittel und Mahlzeiten erstellt werden.

Nahrungsmittel haben einen Namen und bestehen anteilig aus den Nährstoffen: Kohlenhydrate, Fett und Proteine (es gibt wohl noch einige mehr, aber der Einfachheit halber beschränken wir uns auf diese drei). Der Konstruktor nimmt den Namen des Nahrungsmittels als String und die Nährstoffanteile als double. Gibt es negative Anteile, oder ergibt die Summe der Anteile ein Wert größer 1, solle eine `IllegalArgumentException` geworfen werden. Für den Namen gibt es eine Getter-Methode. Die Methode `relativeEnergyDensity` soll den relative Energiedichte des Nahrungsmittels in Kilojoule pro Gramm (kJ/g) zurückgeben. Der Berechnung der relativen Energiedichte sollen die Brennwertangaben der EU zugrunde liegen:

`https://de.wikipedia.org/wiki/Physiologischer_Brennwert#Brennwertangaben_in_der_N.C3.A4hrwertkennzeichnung_der_EU`

Nahrungsmittel sollen eindeutig durch ihren Namen bestimmt sein.

Mahlzeiten haben einen Namen und Zutaten. Letztere werden an den Konstruktor als `Map` übergeben, die für jedes enthaltene Nahrungsmittel die Menge in Gramm angibt. Mahlzeiten haben des Weiteren eine Getter-Methode für den Namen und eine Methode `getCalorificValue`, die den Brennwert der Mahlzeit in Kilojoule (kJ) zurückgibt.

Im Skelett zu dieser Aufgabe finden sie die Klasse `FoodTestExamples` mit einigen Beispieltests.

```
package food;

import org.junit.Test;

import java.util.HashMap;
import java.util.Map;

import static org.junit.Assert.*;

public class FoodTestExamples {

    @Test
    public void testMeal() {
        Map<Food, Double> ingredients = new HashMap<>();
        ingredients.put(new Food("Water", 0.0, 0.0, 0.0), 1000.0);
        ingredients.put(new Food("Sugar", 1.0, 0.0, 0.0), 100.0);
        ingredients.put(new Food("Protein", 0.0, 0.0, 1.0), 100.0);

        Meal shake = new Meal("Fitness-Shake", ingredients);
        assertEquals(3400.0, shake.getCalorificValue(), 0.0001);
    }

    @Test(expected = IllegalArgumentException.class)
    public void testIllegalFood() {
        new Food("Burger", 1, 1, 1);
    }
}
```

Das Beispiel demonstriert auch, wie Sie in JUnit auf das Werfen von Exceptions testen können.

Hinweise

- Die Methoden `equals` und `hasCode` werden von den Collectionklaseen (und überall sonst in Java) verwendet um festzustellen, ob zwei Objekte als gleich zu betrachten sind. Der Gleichheitsoperator (`==`) ist *nur bei primitiven Datentypen* geeignet; für Objekte ist er oft unbrauchbar. Unter diesem Link finden Sie Beispiel zum Implementieren von `equals` und `hashCode`.

  `https://www.mkyong.com/java/java-how-to-overrides-equals-and-hashcode/`

In IntelliJ können Sie sich die Methoden auch über `Code -> Generate` generieren lassen.

- Achtung! Die Gleichheit von Nahrungsmitteln ist in dieser Aufgabe spezifiziert; bitte beachten Sie dies.
- Es ist auch dringend zu empfehlen eine `toString` Methode zu implementieren. Diese wird in Java (und vor allem auch von JUnit) dafür verwendet um Informationen über Objekte in Fehlermeldungen anzuzeigen.