Aufgabenzettel 6

Schnittstellen, abstrakte Klassen, Generics

Dieser Aufgabenzettel muss spätestens in Ihrer Praktikumsgruppe in **KW 23** vorgestellt werden.

Alle für diesen Aufgabenzettel entwickelten Programme sollen sich in einem Paket **de.hsruhrwest.oop.ss2025.assignment6** oder entsprechenden **Unterpaketen** befinden.

Aufgabe 1: Klassenhierarchie mit Schnittstellen

Implementieren Sie die folgende Klassenhierarchie.

- Es gibt eine abstrakte Oberklasse für Küchengeräte. Ein Küchengerät hat einen Namen.
- Es gibt konkrete Unterklassen für die folgenden Küchengeräte.
    - Kochlöffel
    - Toaster
    - Handy
    - Küchenwaage
    - Pürierstab
    - Schale

  Jede Klasse sollte mindestens ein spezifisches Attribut besitzen, das die anderen Klassen nicht besitzen.
- Je nach Eigenschaften sollte jede Unterklasse keine, eine oder mehrere der folgenden Schnittstellen implementieren:
    - Eine Schnittstelle für elektronische Geräte, die eine Methode vorschreibt, die den Stromverbrauch zurückgibt.
    - Eine Schnittstelle für batteriegetriebene Geräte, die eine Methode vorschreibt, die die Batterielaufzeit unter Normalbedingungen zurückgibt.
    - Eine Schnittstelle für gefährliche Geräte, die eine Methode vorschreibt, die das Mindestalter der Person zurückgibt, die das Gerät bedienen darf.
- Implementieren Sie alle Klassen mit plausiblen Werten (z.B. verbraucht ein Toaster zwischen 800 und 1500 Watt).

Aufgabe 2: Abstrakte Klassen

Schreiben Sie eine abstrakte Klasse **AbstractStringTransformer**. Die Klasse bietet eine Methode **transform**. Diese erhält ein Array von Strings und transformiert jeden darin enthaltenen String. Für die Transformation wird eine **abstrakte** Methode transformString aufgerufen. Anschließend wird das transformierte Array ausgegeben. Die abstrakte Methode darf von außen nicht aufrufbar sein (nicht public).

Implementieren Sie zwei konkrete Unterklassen:

- **SpaceRemover** implementiert transformString so, dass alle Leerzeichen entfernt werden und das Ergebnis zurückgegeben wird.
- **Reverser** implementiert transformString so, dass der übergebene String umgedreht zurückgegeben wird.

Aufgabe 3: Generischer Knoten

Schreiben Sie eine generische Klasse **Node**. Sie beschreibt einen Knoten in einer einfach verketteten Liste. Die Klasse besitzt zwei generische Parameter T und N, wobei N von Number abgeleitet sein sollte. Die Klasse besitzt nun drei Attribute:

- Einen Inhalt vom Typ T.
- Eine Anzahl vom Typ N.
- Eine Referenz auf einen Vorgängerknoten, der die gleichen Parameter T und N besitzt. Dieses Attribut darf null sein – dann gibt es keinen Vorgängerknoten.

Implementieren Sie einen Konstruktor, Getter und Setter. Implementieren Sie dann eine Methode **print**. Sie gibt den aktuellen und alle Vorgängerknoten aus. Zusätzlich gibt sie die Summe aller Anzahlen der Knoten aus. Ein Hauptprogramm könnte z.B. so aussehen:

```
var nodeA = new Node<String, Integer>("A", 5);
var nodeB = new Node<String, Integer>("B", 2, numberNodeA);
var nodeC = new Node<String, Integer>("C", 1, numberNodeB);
nodeC.print();
```

Knoten C besitzt B als Vorgänger, B besitzt A als Vorgänger. Eine mögliche Ausgabe wäre nun:

```
C (1)
  B (2)
    A (5)
Total: 8.0
```

Die ausgegebene Gesamtzahl 8 ist die Summe aller Anzahlen der Knoten (5 + 2 + 1).

Aufgabe 4: Generische Statistic

Schreiben Sie eine generische Klasse **Statistik**. Sie besitzt einen generischen Typ T und zwei Attribute vom Typ T, die ein häufigstes und ein seltenstes Element in einem Array repräsentieren sollen. Die Klasse soll mindestens einen Konstruktor und eine toString-Methode besitzen.

Schreiben nun eine generische statische Hilfsmethode **getArrayStatistic**. Sie erhält ein Array vom Typ T und bestimmt das häufigste und das seltenste Element dieses Arrays. Das Ergebnis wird als Objekt von Typ **Statistik** zurückgegeben.