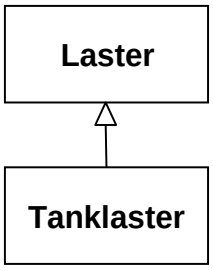
Aufgabe 17.2: Tanklasterverwaltung – Wildcards

Für eine Tanklasterabfüllanlage soll eine Verwaltung für die an der Anlage angemeldeten Laster implementiert werden. Dazu stehen die Dateien `Laster.java` und `Tanklaster.java` als fertig implementierte Klassen bereit, deren Klassenhierarchie die folgende Struktur hat:



Für diese Klassenhierarchie soll eine Utility-Klasse implementiert werden, die nicht volle und volle Laster ermitteln kann. Um die Verwaltung der Laster so allgemein wie möglich zu erlauben, sollen die Methoden der Utility-Klasse gegen die Schnittstelle `Collection<E>` aus der Java-Klassenbibliothek programmiert werden. Die Utility-Klasse soll lediglich die vollen und die nicht vollen Laster in einer Collection ermitteln, an die Verwaltung zurückgeben und den Inhalt einer Collection auf dem Bildschirm ausgeben können. Da die Abfüllanlage in Zukunft vielleicht erweitert und noch andere Tanklasterarten verwaltet werden könnten, sollen die Methoden dieser Utility-Klasse generisch sein.

Von den Methoden der Schnittstelle `Collection<T>` wird in dieser Aufgabe nur die Methode `public boolean add (T obj)` benötigt. Diese Methode `add()` setzt die Referenz vom Typ `T` im Collection-Objekt auf die Referenz, die über `obj` übergeben wurde.

Verwenden Sie für die Implementierung der Utility-Klasse die folgende Vorlage in der Datei `TanklasterUtils_Vorlage.java`. Alle Stellen in der Vorlage, an denen Punkte als Platzhalter stehen, müssen entsprechend ersetzt werden. Bevor Sie die Klasse übersetzen können, müssen Sie die Datei entsprechend umbenennen:

```
// Datei: TanklasterUtils_Vorlage.java

import java.util.Collection;

public class TanklasterUtils
{
   // Statische Methode zum Ermitteln aller nicht vollen Laster in
   // einer Collection. Diese Methode soll als ersten Parameter alle
   // von Collection abgeleiteten Typen akzeptieren, die Referenzen
   // auf Tanklaster-Objekte - oder auf Objekte eines Subtyps –
   // enthalten. Als zweiten Parameter sollen alle von Collection
   // abgeleiteten Typen akzeptiert werden, die Referenzen auf
   // Tanklaster-Objekte – oder auf Objekte eines Basistyps –
   // enthalten.
   public static <T . . . . Collection< . . . . . >,
      R . . . . Collection< . . . . . >>
```

```
      void ermittleNichtVolleLaster
      (T alleLaster, R NichtVolleLaster)
   {
      // Collection der Tanklaster nach nicht vollen Lastern
      // durchsuchen.
      for (Tanklaster l : alleLaster)
      {
         // Prüfe, ob Tanklaster nicht voll.
         if ( . . . . . )
         {
            // Gefundenen Laster zur Collection hinzufügen.
            . . . . . .
         }
      }
   }

   // Statische Methode zum Ermitteln aller vollen Laster in einer
   // Collection.
   // Alle Einschränkungen für Typen, die von der Methode
   // ermittleNichtVolleLaster() akzeptiert werden, gelten auch für
   // diese Methode.
   public static <T . . . . Collection< . . . . . >,
      R . . . . Collection< . . . . . >>
      void ermittleVolleLaster (T alleLaster, R volleLaster)
   {
      // Collection der Tanklaster wird durchsucht nach vollen
      // Lastern
      for (Tanklaster l : alleLaster)
      {
         // Prüfe, ob Laster voll.
         if ( . . . . . )
         {
            // Gefundenen Laster zur Collection hinzufügen
            . . . . . .
         }
      }
   }

   // Statische Methode, um alle Tanklaster in einer Collection
   // auf dem Bildschirm auszugeben. Diese Methode soll als ersten
   // Parameter alle von Collection abgeleiteten Typen akzeptieren,
   // die Referenzen auf Tanklaster-Objekte – oder auf Objekte
   // eines Subtyps - enthalten.
   public static <T . . . . Collection< . . . . . >>
      void printLasterFuellstand (T laster)
   {
      for (Tanklaster l : laster)
      {
         l.print();
         System.out.println ("-----------");
      }
   }
}
```

Zum Testen steht Ihnen die Klasse `TanklasterTest` zur Verfügung:

```
// Datei: TanklasterTest.java

import java.util.ArrayList;
import java.util.Arrays;

public class TanklasterTest
{
   public static void main (String[] args)
   {
      Tanklaster[] lasterArr = new Tanklaster[3];

      // Anlegen dreier neuer Tanklaster
      lasterArr[0] =
         new Tanklaster ("Maimler - Tankstelle Lara", 25000d, 1.5d);
      lasterArr[1] =
         new Tanklaster ("Nam - Tankstelle Muschel", 25000d, 1.7d);
      lasterArr[2] =
         new Tanklaster ("WV - Tankstelle Osse", 2500d, 1.7d);

      // Unterschiedliches Befüllen der angelegten Laster
      lasterArr[0].befuelle (lasterArr[0].getMaxFuellmenge());
      lasterArr[1].befuelle (50000d);
      lasterArr[2].befuelle (1000d);

      // Anlegen einer neuen, leeren Ergebnis-Collection
      ArrayList<Tanklaster> list = new ArrayList<Tanklaster>();

      System.out.println ("Nicht volle Laster:\n");
      // Befüllen der Ergebnis-Collection mit nicht vollen Lastern
      TanklasterUtils.ermittleNichtVolleLaster
         (Arrays.asList (lasterArr), list);
      // Ausgeben der Collection auf dem Schirm
      TanklasterUtils.printLasterFuellstand (list);

      // Säubern der Ergebnis-Collection, um sie wiederzuverwenden
      list.clear();

      System.out.println ("\nVolle Laster:\n");
      // Befüllen der Ergebnis-Collection mit vollen Lastern
      TanklasterUtils.ermittleVolleLaster
         (Arrays.asList (lasterArr), list);
      // Ausgeben der Collection auf dem Schirm
      TanklasterUtils.printLasterFuellstand (list);
   }
}
```