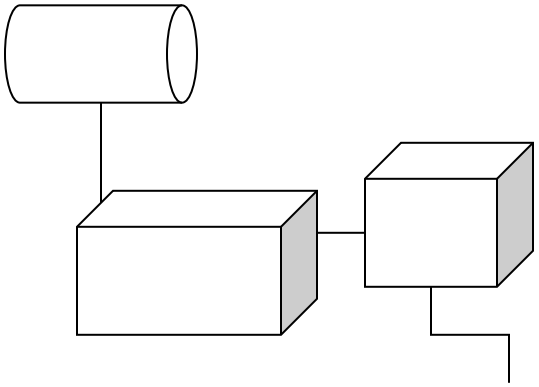
Programmieren 2 (INF)
Übungsblatt 5

Dieses Übungsblatt beschäftigt sich mit objektorientiertem Design, dem nutzbringenden Einsatz von Standard-Containern, dem Zugriff auf Teile eines Ganzen mit Hilfe von Iteratoren, und dem Erstellen tiefer Kopien von polymorphen Strukturen.



Stellen Sie sich vor, ein Freund von Ihnen arbeitet in einem Konstruktionsbüro an der Auslegung von Frischwassertanksystemen. Dabei wählt er in Abhängigkeit von bestimmten Rahmenbedingungen Tanks in verschiedenen Größen und Formen aus und verbindet sie mit Leitungen. Was ihn dabei besonders interessiert, ist das Gesamtvolumen des entstehenden Systems sowie dessen Oberfläche. Letztlich bestimmt die Oberfläche die benötigte Materialmenge sowie das Eigengewicht des Systems.

Die Anforderungen können wie folgt skizziert werden:

- Mit Hilfe des Programms soll ein Modell eines beliebigen geplanten Tanksystems erstellt und analysiert werden können.
- Jeder Tank zeichnet sich durch Oberfläche (Surface) und Volumen (Volume) aus. Diese Werte müssen für alle Tanks berechnet werden können.
- Es gibt sehr unterschiedliche Tanks. Die einfachsten Tanks bestehen genau aus einem Behälter. Davon gibt es drei Typen: quaderförmig (CuboidTank), zylinderförmig (CylindricalTank) und kugelförmig (SphericalTank).
- Verbindet man eine beliebige Anzahl von Tanks mit kleinen Rohren, so entsteht wieder ein Tank mit Oberfläche und Volumen (wobei wir vereinfachend Rohranschlüsse etc. bei der Rechnung ignorieren).
- Ein Tank kann also entweder mit einer der drei Grundformen realisiert, oder aus mehreren Tanks zusammengesetzt werden.

Können Sie Ihrem Freund mit einem kleinen Programm aushelfen?

Aufgabe 1

Erstellen Sie ein erstes Klassendesign in Form eines UML-Modells. Welche Klassen und Interfaces werden für dieses Problem gebraucht? Mit welchen Attributen können die Grundtypen parametrisiert werden? Wie lassen sich die Aufrufe der Berechnungsmethoden für Oberfläche und Volumen bei allen Tanks vereinheitlichen?

Sehen Sie für alle Tank-Implementierungen eine `toString()`-Methode vor. Sie soll Ihnen bei der Visualisierung des erstellten Tanksystem-Modells helfen.

Zusammengesetzte Tanks sollen sich für den Nutzer genauso verhalten wie einfache Tanks. Deshalb sollen alle Tanks mit einem Iterator ausgestattet werden, der das Iterieren über die Teile des Tanks erlaubt, so es denn welche gibt. Bei einfachen Tanks ist das natürlich nicht so spannend, weil der Iterator in der Methode `hasNext()` sofort `false` zurück liefern wird. Wenn nun alle Tanks iterierbar sind, wo sollte diese Eigenschaft in der Klassen- und Interface-Struktur explizit gemacht werden?

Der Lesezugriff kann nun für alle Tanks gleich erfolgen, ohne dass der Nutzer den genauen Typ der Implementierung kennt. Bei der Erzeugung eines Tanksystems, das aus hierarchisch organisierten Tanks besteht, lassen sich die Unterschiede aber nicht so gut verbergen. Sehen Sie für die einfachen Tanks sinnvolle Konstruktoren vor, mit denen ihre Abmessungen parametrisiert werden können. Ein zusammengesetzter Tank soll mit einem parameterlosen Konstruktor erzeugt werden und dann durch Aufrufe einer `add(Tank part)` sukzessive erweitert werden. Welcher Standard-Containertyp eignet sich für die Verwaltung der Teile eines zusammengesetzten Tanks?

Aufgabe 2

Setzen Sie Ihr Design in Java-Code um und implementieren Sie alle notwendigen Methoden. Schreiben Sie auch eine Testmethode, die ein nicht zu triviales Tanksystem aufbaut und dessen Oberfläche und Volumen berechnet.

Aufgabe 3

Erweitern Sie die Tank-Implementierungen um eine `clone`-Methode. Was ist insbesondere bei zusammengesetzten Tanks zu beachten? Testen Sie die Methode, indem Sie

- einen aus mindestens zwei Teilen zusammengesetzten Tank klonen,
- dann im Original über einen Iterator ein Teil löschen,
- und schließlich prüfen, ob die Kopie das gelöschte Teil noch enthält.