

## OOP – Hausaufgabe 02

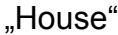

### Thema: Vererbung, Polymorphismus

Aufbau einer Familie von Klassen für geometrische Körper, Untersuchung der Polymorphie-Eigenschaften.

### Vorbereitungen

- Legen Sie ein neues Projekt (als leeres Projekt ohne vorkompilierte Header) an.

### Teilaufgaben

Nr.	Beschreibung	Punkte
1	<p>Entwerfen Sie eine Familie von Klassen für folgende geometrische Figuren:</p> <ul style="list-style-type: none"> <li>• <b>Cube</b> (Würfel),</li> <li>• <b>Cuboid</b> (Quader),</li> <li>• <b>Sphere</b> (Kugel),</li> <li>• <b>Cone</b> (Kreiskegel),</li> <li>• <b>Cylinder</b> (Zylinder),</li> <li>• <b>Prism</b> (allgemeines dreikantiges Prisma)</li> </ul> <p>Ergänzen Sie schließlich die Klassenfamilie auch um die Klassen für die zusammengesetzten Körper:</p> <ul style="list-style-type: none"> <li>• <b>House</b> (liegendes gleichschenkliges Prisma als Dach auf einem Quader gleicher Grundfläche) und</li> <li>• <b>Tower</b> (zylindrischer Turm mit Kegeldach gleicher Grundfläche)</li> </ul> <div style="display: flex; justify-content: space-around; margin-top: 20px;"> <div style="text-align: center;">  <p>„House“</p> </div> <div style="text-align: center;">  <p>„Tower“</p> </div> </div> <p>Alle benötigten Längen sollen als Gleitkommazahlen doppelter Genauigkeit verwaltet werden. Die Längeneinheit spiele dabei zunächst keine Rolle (es können z. B. Meter angenommen werden).</p>	12

Nr.	Beschreibung	Punkte
	<p>Alle Klassen sollen die parameterlosen Methoden</p> <ul style="list-style-type: none"> <li>• <b>double getVolume() const</b></li> <li>• <b>double getSurface() const</b></li> </ul> <p>implementieren, die jeweils die betreffende Gleitkommazahl des Volumens bzw. der äußeren Körperoberfläche in der sich aus der angenommenen Längeneinheit direkt ableitenden Volumen- oder Flächeneinheit liefern. Entwerfen Sie die Klassenhierarchie so, dass alle Klassen polymorph auf eine <u>abstrakte</u> Basisklasse <b>Body</b> zurückführbar sind. Diese sollte die Methoden <b>getVolume</b> und <b>getSurface</b> <i>rein virtuell</i> deklarieren, um zugleich eine Instanziierung von Objekten der Klasse <b>Body</b> zu verhindern.</p> <p>Jedes Objekt der Körperklassen soll einen zwingend vorzugebenden Lage-Referenzpunkt im <math>\mathbb{R}^3</math> besitzen (Klasse <b>Point3D</b>), der per Definition in seinem Volumenschwerpunkt liege und der mit einer Methode</p> <ul style="list-style-type: none"> <li>• <b>Point3D getLocation() const</b></li> </ul> <p>erfragbar sein soll. Die Klasse <b>Point3D</b> selbst soll mit sogenannten Getter-Methoden <b>getX</b>, <b>getY</b> und <b>getZ</b> ausgestattet sein.</p> <p>Stellen Sie für jede konkrete Körperklasse mindestens einen geeigneten Konstruktor bereit, mit dem der Körper beim Instanzieren in seinen Abmaßen und seiner Lage eindeutig bestimmt wird. Die Konstruktoren sollten dazu nur so viele Parameter wie nötig haben (keine inkonsistenten Werte-Tupel zulassen)!</p> <p>Implementieren Sie die angegebenen Methoden.</p>	
2	<p>Legen Sie in der Basisklasse <b>Body</b> eine rein virtuelle Methode <b>getBodyName()</b>, die eine Zeichenkette zurückgibt. Die konkreten Körperklassen sollen dann jeweils eine Bezeichnung der Körperart zurückgeben, z. B. „Würfel“ oder „cube“ bei Klasse <b>Cube</b>. Sorgen Sie dafür, dass sich jeweils das Objekt durch den Methodenaufruf nicht verändern kann.</p>	3
3	<p>Erstellen Sie eine Klasse <b>BodyQueue</b>, die eine unbeschränkte Warteschlange von beliebigen Körpern der in Teilaufgabe 1 erstellten Klassen darstellen soll. Achten Sie beim Entwurf der Klasse darauf, dass die polymorphen Eigenheiten der konkreten Elementkörper bei deren Einlagerung nicht verloren gehen.</p> <p>Die Signaturen der Methoden zum Anfügen bzw. Entnehmen eines Körpers aus der Warteschlange seien:</p> <pre><b>void enqueue(const Body&amp; body);</b> <b>const Body* dequeue();</b></pre> <p>Erstellen Sie ferner eine Methode, die feststellt, ob die Schlange leer ist.</p>	5
4	<p>Erstellen Sie zwecks Einheitlichkeit nun in allen Körperklassen (außer <b>Body</b> selbst) jeweils eine <u>statische</u> (Klassen-) Methode <b>Body* createByDialog()</b>, die der Erzeugung eines neuen Objekts der jeweiligen Klasse <i>im Dialog</i> dient, wobei die Methode <b>createByDialog()</b> selbst die für den</p>	10

Nr.	Beschreibung	Punkte
	Konstruktoraufruf benötigten Werte vom Nutzer abfragen soll. (Dies bündelt die Kompetenz zur Interpretation des Konstruktors bei der jeweiligen Klasse selbst.)	
5	<p>Schreiben Sie nun ein Hauptprogramm, das eine Körperwarteschlange entsprechend Teilaufgabe 3 verwaltet und dem Nutzer in einer Schleife per Menü folgende Optionen anbieten soll:</p> <ul style="list-style-type: none"> <li>• Erzeugung eines Körpers einer vom Nutzer wählbaren Körperart. Das erzeugte Objekt soll dabei an die Warteschlange angehängt werden.</li> <li>• Entnahme eines Objekts aus der Warteschlange. (Diese Option soll nur angeboten werden, wenn die Warteschlange nicht leer ist.) Für das entnommene Objekt sollen dann folgende Angaben ausgegeben werden, anschließend ist der Körper zu beseitigen: <ul style="list-style-type: none"> <li>○ Name der Körperart;</li> <li>○ Lage des Schwerpunkts;</li> <li>○ Volumen;</li> <li>○ Oberfläche.</li> </ul> </li> <li>• Verlassen des Programms.</li> </ul> <p>Tipp: Machen Sie sich zum Test beim Einlagern in die Warteschlange Notizen über die erzeugten Körper und deren Maße, damit Sie die Ergebnisse überprüfen können.</p>	5
*6	<p><i>Freiwillige Zusatzaufgabe:</i></p> <p>Sorgen Sie durch einen geeigneten Mechanismus dafür, dass jeder einzelne Körper bei seiner Erzeugung automatisch eine fortlaufende eindeutige (und unveränderliche, aber öffentlich lesbare) Nummer (als „Id“) erhalten. Die Id-Vergabe soll unabhängig von der Körperart erfolgen.</p> <p>(Beschreiben Sie, wie die Lösung verändert werden müsste, wenn die Id-Vergabe für jede Körperart separat erfolgen sollte? Probieren Sie es aus.)</p> <p>Lassen Sie das Hauptprogramm beim Entnehmen eines Elements auch dessen Id mit ausgeben.</p>	(5)
*7	<p><i>Freiwillige Zusatzaufgabe:</i></p> <p>Erstellen Sie (analog zum Traffic-Projekt) eine Klasse <b>Length</b>, deren Instanzen jeweils einheitenbehaftete Längenwerte repräsentieren sollen und sowohl in einer vorgegebenen Längeneinheit erstellt werden als auch – mittels einer Methode <b>getValue</b> die Maßzahl des Längenobjekts bezogen auf eine vorgegebene Längeneinheit liefern können (Default-Einheit soll jeweils Meter sein, es sollen mindestens je drei metrische und imperiale Einheiten verfügbar sein). Modifizieren Sie die Konstruktoren der <b>Body</b>-Klassen so, dass die Längenwerte als <b>Length</b>-Objekte vorgegeben werden können, und die Methoden <b>getSurface</b> und <b>getVolume</b> so, dass optional eine Längeneinheit vorgebar ist und das Ergebnis dann als Maßzahl für die jeweils zur Längeneinheit passende Flächen- bzw. Volumeneinheit zurückgegeben wird. Ändern Sie auch die <b>Point3D</b>-Klasse so, dass sie wahlweise</p>	(10)

Nr.	Beschreibung	Punkte
	entweder mit drei unabhängigen <b>Length</b> -Objekten oder mit drei <b>double</b> -Werten und einer gemeinsam geltenden optionalen Längeneinheit konstruierbar sind; die Koordinaten-Getter-Methoden sollen ebenfalls eine Längeneinheit als optionalen Parameter erhalten.	