FAKULTÄT FÜR INFORMATIK
Faculty of Informatics

Objektorientierte Programmiertechniken
LVA 185.A01, VL 2.0, 2011 W

3. Übungsaufgabe

Themen:
Untertypbeziehungen, Zusicherungen

Termine:

Ausgabe:	09.11.2011
reguläre Abgabe:	16.11.2011, 13:45 Uhr
nachträgliche Abgabe:	23.11.2011, 13:45 Uhr

Abgabeverzeichnis:
Gruppe/Aufgabe3

Programmaufruf:
java Test

Grundlage:
Skriptum bis Seite 71 sowie 90 bis 92

Aufgabe

Welche Aufgabe zu lösen ist:
Folgendes Interface ist vorgegeben:

```
public interface Polygon { // maybe not regular
   int edges();        // number of edges >= 3
   double area();      // > 0.0 (Flächeninhalt)
   double perimeter(); // > 0.0 (Umfang)
}
```

Es sollen folgende Typen (also Klassen, abstrakte Klassen oder Interfaces) als Untertypen von *Polygon* erstellt werden:

- Die Seitenlängen der Instanzen von *Rectangle* (Rechteck) werden im Konstruktor gesetzt. Änderungen der Seitenlängen sind jederzeit möglich, aber nur so, dass das Verhältnis zwischen den Seitenlängen gleich bleibt. Zu diesem Zweck gibt es die Methode *scale*, welche die Seitenlängen mit dem als Parameter übergebenen Faktor multipliziert.
- Die Seitenlänge von *Square* (Quadrat) wird im Konstruktor gesetzt und Änderungen der Seitenlänge sind jederzeit über die Methode *set* möglich. Außerdem gibt es die Methode *scale* zum Skalieren der Seitenlänge.
- Die drei Seitenlängen von *Triangle* (Dreieck) werden im

Konstruktor gesetzt und sind jederzeit getrennt voneinander über die Methoden *setA*, *setB* und *setC* änderbar.
- Die Seitenlänge von *EquilateralTriangle* (gleichseitiges Dreieck) wird im Konstruktor gesetzt und Änderungen der Seitenlänge sind jederzeit über die Methode *set* möglich. Außerdem gibt es die Methode *scale* zum Skalieren der Seitenlänge.
- Alle Seiten und Winkel in Instanzen von *RegularPolygon* (regelmäßiges Vieleck) sind gleich. Es gibt die Methoden *set* zum Setzen und *scale* zum Skalieren der Seitenlänge.

Versehen Sie (abstrakte) Klassen und Interfaces mit allen notwendigen Zusicherungen (entsprechend obigen Beschreibungen) und stellen Sie sicher, dass Sie nur dort eine Vererbungsbeziehung (*extends* oder *implements*) verwenden, wo tatsächlich eine Untertypbeziehung (auch hinsichtlich der Zusicherungen) besteht. Zwischen je zwei Untertypen von *Polygon* soll eine Untertypbeziehung bestehen, wenn dies aufgrund der Schnittstellen und Zusicherungen möglich ist.

Sorgen Sie dafür, dass alle Methoden, die dasselbe Verhalten aufweisen, mit möglichst wenig Wissen über die Klasse eines Objekts aufrufbar sind. Sie können zu diesem Zweck zusätzliche (abstrakte) Klassen und Interfaces einführen, die Sie als vorteilhaft erachten. Die Typstruktur soll trotzdem möglichst einfach und klein bleiben, wobei jedoch alle oben genannten Typen (mit den vorgegebenen Namen) vorkommen müssen, auch solche, die Sie vielleicht für nicht nötig erachten.

Schreiben Sie eine Klasse *Test* zum Testen Ihrer Lösung. Erzeugen Sie Instanzen aller oben genannter Typen, wobei aufgrund von Polymorphismus ein einziges Objekt natürlich auch mehrere Typen abdecken kann. Überprüfen Sie so gut Sie können mittels Testfällen, ob dort, wo Sie eine Untertypbeziehung annehmen, Ersetzbarkeit gegeben ist. Wenn zwischen zwei der oben beschriebenen Typen keine Untertypbeziehung besteht, geben Sie in einem Kommentar in der Testklasse ein kurzes Beispiel an, in dem die Ersetztbarkeit verletzt ist.

Wie die Aufgabe zu lösen ist:
Anzahl und Umfang der benötigten Interfaces und Klassen sind überschaubar. Die Schwierigkeit liegt darin, *alle* Untertypbeziehungen zu finden und Ersetzbarkeit sicherzustellen. Dieser Punkt ist wesentlich für die Beurteilung. Vererbungsbeziehungen, die nicht gleichzeitig auch Untertypbeziehungen sind, führen zu sehr hohen Punkteabzügen. Ebenso gibt es hohe Punkteabzüge für nicht wahrgenommene Gelegenheiten, Untertypbeziehungen zwischen den Untertypen von *Polygon* herzustellen. Nennenswerte Abzüge gibt es auch in Fällen, in denen oben genannte Methoden nicht so allgemein aufrufbar sind, wie dies möglich wäre, oder wenn Zusicherungen mangelhaft sind. Die direkte Codewiederverwendung durch Vererbung spielt für die Beurteilung dagegen keine Rolle; Methoden dürfen also beliebig oft implementiert sein.

Eine Grundlage für das Auffinden der Untertypbeziehungen sind gute

Zusicherungen. Wesentliche Zusicherungen kommen bereits in obigen Beschreibungen der benötigten Typen vor. Sie brauchen die einzelnen Beschreibungsteile nur mehr richtig zuordnen. Zusätzlich sind vielleicht Bedingungen auf Parametern nötig, um die Zusicherungen in *Polygon* zu erfüllen. Untertypbeziehungen ergeben sich aus den erlaubten Beziehungen zwischen Zusicherungen in Unter- und Obertypen. Es hat sich als günstig erwiesen, alle Zusicherungen, die in einem Obertyp gelten, im Untertyp direkt bei den betroffenen Methoden nochmals hinzuschreiben, da sie sonst leicht übersehen werden.

Vergewissern Sie sich der Korrektheit der Untertypbeziehungen zusätzlich über geeignete Testfälle. Die Anzahl der Testfälle ist nicht entscheidend, wohl aber deren Qualität: Es kommt darauf an, dass die Testfälle mögliche Verletzungen der Ersetzbarkeit aufdecken können. Umgekehrt sollen Sie sich auch vergewissern, dass Sie keine Gelegenheit für Untertypbeziehungen verpasst haben, indem Sie Beispiele dafür finden, wie angenommene Untertypbeziehungen das Ersetzbarkeitsprinzip verletzen würden. Schreiben Sie die Gegenbeispiele als Kommentare neben den Testfällen für Ersetzbarkeit in die Testklasse.

Zusicherungen in Testklassen werden aus praktischen Überlegungen bei der Beurteilung nicht berücksichtigt. Sorgen Sie aber bitte dafür, dass ein Aufruf von *java Test* einigermaßen nachvollziehbaren Output generiert.

Zur Lösung dieser Aufgabe müssen Sie Untertypbeziehungen und vor allem den Einfluss von Zusicherungen auf Untertypbeziehungen im Detail verstehen. Holen Sie sich entsprechende Informationen aus Kapitel 2 des Skriptums. Folgende zusätzlichen Informationen könnten hilfreich sein:

- Konstruktoren werden in einer konkreten Klasse aufgerufen und sind daher vom Ersetzbarkeitsprinzip nicht betroffen. Konstruktoren haben wie statische Methoden keinen direkten Einfluss auf Untertypbeziehungen.
- Zur Lösung der Aufgabe benötigen Sie keine Exceptions. Sollten Sie dennoch Exceptions verwenden, bedenken Sie, dass eine Instanz eines Untertyps nur dann eine Exception werfen darf, wenn man auch von einer Instanz eines Obertyps in derselben Situation diese Exception erwarten würde.
- In der Aufgabe benötigen Sie auch keine Generizität. Sollten Sie dennoch Generizität verwenden, bedenken Sie, dass sich viele Warnungen des Compilers im Zusammenhang mit Generizität auf Verletzungen der Ersetzbarkeit beziehen und daher schwerwiegende Fehler im Sinne dieser Aufgabe darstellen. Vermeiden Sie daher diesbezügliche Meldungen des Compilers.

Lassen Sie sich von der Form der Beschreibung der benötigten Typen nicht täuschen. Daraus, dass die Beschreibung eines Typs die Beschreibung eines anderen Typs teilweise wiederholt, folgt noch keine Ersetzbarkeit. Generell sind Sie wahrscheinlich auf dem falschen Weg,

wenn es den Anschein hat, A könne Untertyp von B und B gleichzeitig Untertyp von A sein, obwohl A und B ungleich sind.

Achten Sie auf richtige Sichtbarkeit. Alle oben beschriebenen Typen und Methoden sollen überall verwendbar sein. Die Sichtbarkeit von Implementierungsdetails und insbesondere von Variablen soll aber so stark wie möglich eingeschränkt werden.

Was man generell beachten sollte:
Es werden keinerlei Ausnahmen bezüglich des Abgabetermins gemacht. Was nicht rechtzeitig im richtigen Verzeichnis am Übungsrechner steht, wird bei der Beurteilung nicht berücksichtigt. Da es ab jetzt um 100 Punkte pro Aufgabe geht, wäre jede Ausnahme anderen Gruppen gegenüber ungerecht.

Bitte verwenden Sie in dieser und den folgenden Aufgaben keine Unterverzeichnisse im Abgabeverzeichnis (und damit auch keine Pakete). Das Verbot von Unterverzeichnissen hat sich zur Vermeidung vielfältiger Probleme bei der Abgabe und Beurteilung bewährt.

Schreiben Sie nicht mehr als eine Klasse in jede Datei (ausgenommen geschachtelte Klassen), halten Sie sich an übliche Namenskonventionen in Java (Großschreibung für Namen von Klassen und Interfaces, kleine Anfangsbuchstaben für Variablen und Methoden, etc.), und verwenden Sie die Namen, die in der Aufgabenstellung vorgegeben sind. Damit erhöhen Sie die Lesbarkeit Ihrer Programme ganz wesentlich. Außerdem können derartige “Fehler” zu Punkteverlusten führen.

Übernehmen Sie das vorgegebene Interface *Polygon* bitte unverändert. Sie müssen die Datei selbst erzeugen, da im Abgabeverzeichnis vorinstallierte Dateien leicht zu Problemen bei der Abgabe führen können.

Warum die Aufgabe diese Form hat:
Im Gegensatz zu vielen anderen Aufgaben ist diese Aufgabe ziemlich klar spezifiziert. Der Grund liegt darin, dass die Beschreibungen der Typen keinen Interpretationsspielraum bezüglich der Ersetzbarkeit bieten sollen. Die Aufgabe ist so formuliert, dass die Untertypbeziehungen (abgesehen von Typen, die Sie vielleicht zusätzlich einführen) eindeutig sind. Über Testfälle und Gegenbeispiele in Kommentaren sollten Sie in der Lage sein, große Fehler in der Struktur der Lösung selbst zu finden. Eine Voraussetzung für das Erkennen der richtigen Lösung und deren Eindeutigkeit ist aber ein gutes Verständnis der Ersetzbarkeit. Wenn Ihnen mehrere Lösungsmöglichkeiten (hinsichtlich der Struktur ohne eigene Typen) als gleichermaßen richtig erscheinen, sollten Sie nocheinmal ins Skriptum schauen.

Anfang | HTML 4.01 | letzte Änderung: 2011-11-09 (Puntigam)