TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Praktikum Grundlagen der Programmierung
Prof. Dr. Helmut Seidl

SS 2008
Aufgabenblatt 6
26. Mai 2008

Abgabe: 03.-05. Juni 2008 beim jeweiligen Tutor

Praktikum Grundlagen der Programmierung

Themen: Vererbung, abstrakte Klassen

Wichtiger Hinweis: Es wird dringend angeraten, die oben genannten Kapitel im Skript *vor* dem Praktium *selbständig* zu erarbeiten, um Unklarheiten, insbesondere im Bezug auf die Hausaufgaben, rechtzeitig mit dem Tutor abklären zu können.

Aufgabe 31 (Ü) Modellierung

In dieser Aufgabe soll ein Klassenmodell für Mietwohnungen implementiert werden, das es erlaubt den Mietpreis einer Mietwohnung zu bestimmen. Eine Mietwohnung besteht aus mehreren Räumen, die eine Länge und Breite in Metern haben. Die Grundmiete beträgt 10 EUR pro m^2. Wir unterscheiden zwei Arten von Räumen:

- **Wohnräume:** können möbliert sein. Die Möblierung hat einen Wert. Der Mietpreis erhöht sich durch eine Möblierung um 1 % des Möblierungswerts.
- **Funktionsräume:** können einen Wasseranschluss haben. Ist dies der Fall, verdoppelt sich die Grundmiete für den Raum.

a) Definieren Sie die Klasse `Raum`. Die Werte für Länge und Breite des Raumes sollen im Konstruktor gesetzt werden können. Der Zugriff auf die Attribute soll zusätzlich über entsprechende `get`- und `set`-Methoden ermöglicht werden. Weiterhin soll die Klasse eine Methode **`public int`** `berechneMietpreis()` anbieten, die die Grundmiete des Raumes berechnet.

b) Erweitern Sie Ihr Klassenmodell entsprechend der Beschreibung aus der Aufgabenstellung um die beiden Klassen `Wohnraum` und `Funktionsraum`. Nutzen Sie in Ihrer Implementierung möglichst geschickt die Konzepte von Objektorientierung und Vererbung aus

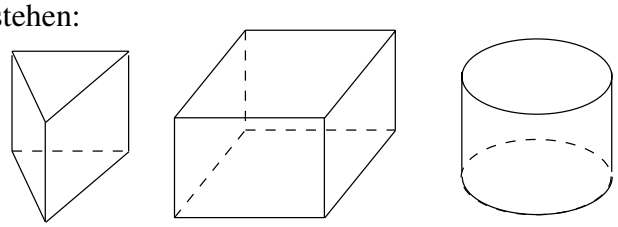
c) Definieren Sie die Klasse `Mietwohnung`.

Die Klasse soll eine Methode **`public void`** `fuegeRaumHinzu(Raum r)` anbieten, die einen Raum `r` zur Mietwohnung hinzufügt. Unmittelbar nach der Erzeugung eines `Mietwohnung`-Objekts soll die Mietwohnung leer sein. **Hinweis:** Verwenden Sie in Ihrer Implementierung die unten angegebene generische Klasse `List`.

d) Erweitern Sie die Klasse `Mietwohnung` um eine Methode **`public int`** `berechneMietpreis()`. Als Ergebnis soll die Methode die Summe der Mietpreise der einzelnen Räume der Mietwohnung zurückliefern.

Aufgabe 32 (Ü) Vererbung

Prismen oder *Zylinder* sind geometrische Körper, die durch Parallelverschiebung ihrer Grundfläche im Raum entstehen. Die folgende Abbildung zeigt als Beispiele für Prismen ein Dreieckprisma, einen Quader und einen Zylinder, die durch Verschiebung eines Dreiecks, eines Rechteckes bzw. eines Kreises entstehen:



Würfel sind durch ihre Höhe eindeutig bestimmt.

Quader sind durch ihre Höhe, Länge und Breite bestimmt.

Kreiszylinder sind durch ihre Höhe und den Radius ihrer Grundfläche bestimmt.

regelmässige Vieleckprismen sind durch ihre Höhe und die Seitenlänge ihrer jeweiligen Grundfläche bestimmt. In Frage kommen zum Beispiel

- *regelmässige Dreieckprismen*;
- *quadratische Quader* oder
- *regelmässige n-Eckprismen*.

An Operationen sollen die Körper die Berechnung von Umfang und Flächeninhalt ihrer *Grundfläche*, die Berechnung ihrer *Mantelfläche*, *Oberfläche* und des *Volumen*s zur Verfügung stellen, sowie den *Vergleich* ihrer Volumina mit anderen geometrischen Körpern unterstützen.

In dieser Aufgabe soll eine Klassenhierarchie für diese geometrischen Körper in UML modelliert und in Java implementiert werden.

a) Geben Sie ein Klassendiagramm zur Modellierung der oben genannten Körper in UML an.
b) An welcher Stelle Ihrer Klassenhierarchie müssen die in a) spezifizierten Operationen implementiert werden, damit möglichst viel in Unterklassen wiederverwendet werden kann?
c) Greifen Sie 3 geometrische Körper aus Ihrem Klassenmodell heraus und implementieren Sie diese in Java. Schreiben Sie zudem ein kleines Testprogramm.

Hinweise:

- Fassen Sie gleichartige Attribute und Methoden in einer geeigneten Oberklasse zusammen.
- Die Fläche eines gleichseitigen Dreiecks mit Seitenlänge a ist $\frac{a^2}{4}\sqrt{3}$
- Die Fläche eines regelmässigen Sechsecks mit Seitenlänge a ist $\frac{3a^2}{2}\sqrt{3}$.
- Die Java-Klasse `Math` stellt in der Klassenvariablen `Math.PI` einen Wert für π sowie eine Klassenmethode `Math.sqrt()` zur Berechnung der Quadratwurzel zur Verfügung.

Aufgabe 33 (H) Die große Schleife **(20 Punkte)**

Ziel dieser Aufgabe ist es, ein großes Radrennen mit verschiedenen beteiligten Personen (Fahrertypen, Kontrolleure, Ärzte) in unterschiedlicher Geländelage objektorientiert zu modellieren und implementieren.

a) Erstellen Sie zunächst die Klassenhierarchie zur Repräsentation der beteiligten Personen. Beginnen dazu Sie mit der abstrakten Basisklasse `Person`. Diese soll Attribute zur Speicherung von Name, Vorname und Geburtsdatum besitzen.

b) Fahren Sie mit der Klasse `Kontrolleur` fort. Ein Objekt der Klasse `Kontrolleur` hat zusätzlich ein Attribut zur Speicherung der bereits ertappten Rennfahrer.

c) Objekte der Klasse `Arzt` unterscheiden sich von Objekten der Klasse `Person` durch Ihren Erfahrungwert, welcher als Zahl zwischen 0 und 1 zu modellieren ist.

d) In der abstrakten Basisklasse `Rennfahrer` sollen die für alle Fahrertypen gemeinsamen Attribute und Methoden festgehalten werden. Beachten Sie, dass ein Fahrer auch eine Person ist. Ein Fahrer hat die folgenden allgemeinen Merkmale:

- aktuelle Geschwindigkeit in km/h
- seine Fitness (zu modellieren als Zahl zwischen 0 und 1)
- seinen Fairnessfaktor (zu modellieren als Zahl zwischen 0 und 1), der angibt, mit welcher Wahrscheinlichkeit ein Fahrer nach Test durch einen Kontrolleur nicht disqualifiziert wird.
- seinen(`Arzt`)

e) Implementieren Sie eine eine Methode **`void`** `arztBesuchen()`, welche die Fitness eines Fahrers vor der nächsten Etappe verbessert. Bei einem absolut fairen Fahrer verbessert sich die Fitness um ein Hundertstel der Erfahrung des Arztes. Bei einem unfairen Sportler beträgt die Verbesserung zwei Hundertstel, jedoch wird auch sein Fairnessfaktor um den gleichen Betrag verkleinert. (Negative Werte sind nicht möglich.)

f) eine Methode **`abstract double`** `bewaeltigeEtappe(Etappe e)`, Konkrete Implementierungen der Unterklassen sollen für jeden Fahrer die Zeit in Stunden zurückgeben, die er zur Bewältigung der Etappe `e` benötigt und seinen aktuellen Fitnesszustand um ein Zehntausendstel der Länge der aktuell zurückgelegten Teilstrecke verkleinern.

g) Implementieren Sie die konkreten Fahrertypen: `Kletterer`, `Sprinter`, `Zeitfahrer`. Die Vorschriften zur Berechnung der Durchschnittsgeschwindigkeit können Sie der folgenden Tabelle entnehmen:

Fahrertyp	v_{max}	$v_{BergEtappe}$	$v_{FlachEtappe}$	$v_{ZeitfahrEtappe}$
Kletterer	35km/h	$v_{max} \cdot f$	$v_{max} \cdot \sqrt[5]{f}$	$v_{max} \cdot \sqrt[4]{f}$
Zeitfahrer	40km/h	$v_{max} \cdot f^2$	$v_{max} \cdot \sqrt[2]{f}$	$v_{max} \cdot f$
Sprinter	45km/h	$v_{max} \cdot f^3$	$v_{max} \cdot f$	$v_{max} \cdot \sqrt[2]{f}$

(v_{max} bezeichnet die maximale Geschwindigkeit eines Fahrertyps, f die aktuelle Fitness)

Nutzen Sie bei Ihrer Implementierung Vererbung möglichst gut aus.

h) Zur Modellierung der Beschaffenheit einer zu bewältigenden Strecke, sollen Sie die folgenden Geländetypen der Etappen implementieren:

- `BergEtappe`
- `ZeitfahrEtappe`
- `FlachEtappe`

Jede Etappe ist mit einer `laenge` (in km) versehen.

i) Jede der zu Implementierenden Klassen soll auch eine `toString()`-Methode besitzen, welche eine textuelle Beschreibung des jeweiligen Objektes zurückliefert. Verwenden und Überschreiben Sie dazu in sinnvoller Weise die gleichnamigen Methoden der Oberklassen.

j) Implementieren Sie die Klasse `Radrennen`, bei der ein Radrennen auf einer bestimmten Gesamtstrecke simuliert wird. Setzen Sie dazu in der `main`-Methode die Rennstrecke, als Sequenz von o.g. Teilstrecken fest. An diesem Wettrennen nehmen je zwei Fahrer pro Fahrtypen teil, wovon einer mit einem Fairnesswert von 1.0 und der andere mit einer Zufallszahl initialisiert wird. Direkt im Anschluß an eine Etappe testet der Kontrolleur einen zufällig ausgewählten Fahrer. Ausserdem nimmt jeder Fahrer nach jeder Etappe eine medizinische Leistungen bei seinem Arzt in Anspruch, deren Erfahrungswert zu Beginn der Simulation ebenfalls auf einen Zufallswert gesetzt wird.

   Am Ende des Rennens soll der Sieger der Rundfahrt – der für die gesamte Strecke die wenigste Zeit gebraucht hat – ermittelt und ausgegeben werden.

Anmerkung: Verwenden Sie während der Implementierung Testausgaben, die in der endgültigen Version Ihres Programmes auskommentiert werden. Die Fehlersuche wird damit deutlich einfacher!