Übungszettel, 2015-05-11

Abgabe möglich bis Donnerstag 2015-06-11T2359
Speichern Sie Ihre Lösungen in Ihrem Ordner im Repository ab, bevor Sie sie vorführen und erklären.

33. Interfaces [1+1+3 Punkte]

33.1. Schreiben Sie ein Programm, das folgende Interfaces definiert: Position (ohne Methoden), Vehicle (mit Methoden int getMaxSpeed() und moveTo(Position newPosition)), Car (drive()), Aircraft (fly()).

33.2. Erweitern Sie Ihr Programm um die Definition einer Klasse Porsche, die das Interface Car implementiert, einer Klasse Airbus, die das Interface Aircraft implementiert, und einer Klasse Ferry, die das Interface Vehicle implementiert. Ergänzen Sie eine Klasse Zeppelin, die die Interfaces Aircraft und Car implementiert.

33.3. Schreiben Sie ein Programm, das folgende Interfaces definiert: GeometricObject (getX(), setX(), getY(), setY(), setColour(MyColour), getColour()), Circle (getX(), setX(), getY(), setY(), setColour(MyColour), getColour()). MyColour soll ein enum-Datentyp sein. Definieren Sie in Ihrem Programm für jedes Interface zwei Klassen, die jeweils das gleiche Interface implementieren, ohne dass die Klassen in einer Beziehung zueinander stehen. Sie dürfen in dieser Teilaufgabe weder für die Interfaces noch für die Klassen Vererbung einsetzen.

34. Abstrakte Klassen [3 Punkte]

Schreiben Sie zu jedem Interface der vorherigen Teilaufgaben eine abstrakte Klasse, die alle vom Interface definierten Methoden ohne Inhalt und gekennzeichnet als abstrakte Methoden enthält. Ergänzen Sie zu jeder abstrakten Klasse eine abgeleitete Klasse, die nicht mehr abstrakt ist.

35. Konstruktoren [1+1 Punkte]

35.1. Schreiben Sie ein Programm, das drei Klassen enthält, die alle voneinander abgeleitet sind: Broccoli als Unterklasse von Vegetable und Vegetable als Unterklasse von Food. Geben Sie im Konstruktor jeder Klasse aus, wie die Klasse heißt. Erzeugen Sie ein Objekt der Klasse Broccoli. Welche Ausgabe erhalten Sie und warum?

35.2. Erweitern Sie die vorhergehende Teilaufgabe, sodass jede Klasse zwei Konstruktoren besitzt: einen parameterlosen und einen Konstruktor, der einen String als Parameter erhält. Der parameterbehaftete Konstruktor soll den übergebenen String auf der Konsole ausgeben. Erzeugen Sie ein Objekt der Klasse Broccoli, einmal mit dem parameterlosen Konstruktor und einmal mit dem parameterbehafteten Konstruktor. Welche Ausgabe erhalten Sie und warum?

36. Überladen von Methoden [1+1 Punkte]

36.1. Schreiben Sie ein Programm, das ein Interface IAddress (mit Methoden get/setStreet(), get/setPostcode(), get/setCity()) definiert. Für setCity() soll es möglich sein, wahlweise nur den Ort oder den Ort zusammen mit der Postleitzahl oder nur die Postleitzahl anzugeben. Implementieren Sie das Interface IAddress mit einer Klasse Address.

36.2. Schreiben Sie ein Programm, das eine Klasse UniversalOutput definiert mit der Methode printToScreen(). An printToScreen() soll man wahlweise int, double, Char, String, Object übergeben können. Der Wert des übergebenen Parameters soll auf die Konsole ausgegeben warden.

37. Überschreiben von Methoden [1+1 Punkte]

37.1. Schreiben Sie ein Programm, das drei Klassen enthält, die alle voneinander abgeleitet sind: Broccoli als Unterklasse von Vegetable und Vegetable als Unterklasse von Food. Broccoli und Food sollen eine Methode eat(int) enthalten, die den Namen der Klasse ausgibt. Vegetable soll eine Methode eat(double) enthalten, die den Namen der Klasse ausgibt. Erzeugen Sie ein Objekt der Klasse Broccoli und rufen Sie für das Objekt eat(42) und eat (42.0) auf. Welche Ausgabe erhalten Sie und warum?
Warum sollten Sie Ihre Methoden in Unterklassen mit dem Schlüsselwort @Override kennzeichnen?

37.2. Ergänzen Sie Broccoli.eat() um einen Aufruf von super.eat() zu Beginn der Methode. Welche Ausgabe erhalten Sie und warum?

38. Vererbung (i) – Subtyping [2+1 Punkte]

38.1. Schreiben Sie ein Programm, das folgende Interfaces definiert:

- Vehicle
- Car, abgeleitet von Vehicle
- Aircraft, abgeleitet von Vehicle
- Boat, abgeleitet von Vehicle
- Ufo

Definieren Sie folgende Klassen:

- Porsche, implementiert Car
- VW, implementiert Car
- Airbus, implementiert Aircraft
- Ferry, implementiert Boat,
- Seaplane, implementiert Aircraft und Boat
- Tricycle, implementiert Vehicle
- FlyingSaucer, implementiert Ufo und Aircraft

Erzeugen Sie ein Object-Array mit den Komponenten new Porsche, new VW, new Airbus, new Ferry, new Seaplane, new Tricycle, new FlyingSaucer. Geben Sie die Komponenten des Arrays aus, indem Sie for-each-Schleifen einsetzen, in denen Sie mittels instanceof filtern, sodass nur Komponenten eines bestimmten Typs ausgegeben warden. Fügen Sie nach jeder Schleife eine Leerzeile in die Ausgabe ein. Geben Sie so alle Vehicle-, alle Car-, alle Aircraft- und alle Boat-Objekte aus.

38.2. Ergänzen Sie das Programm, indem Sie einer Aircraft-Variable ein Seaplane-Objekt zuweisen. Weisen Sie nun einer Boat-Variable ein neues Seaplane-Objekt zu. Weisen Sie der Boat-Variable dann den Wert der Aircraft-Variable zu. Was müssen Sie tun, damit dies funktioniert?

39. Vererbung (ii) – Subclassing [1+5 Punkte]

39.1. Definieren Sie die VW-Klasse als Unterklasse der Porsche-Klasse und die Seaplane-Klasse als Unterklasse der Airbus-Klasse. Was müssen Sie beachten, wenn Sie die Seaplane-Klasse als Unterklasse der Ferry-Klasse definieren wollen?

39.2. Ändern Sie das Postleitzahlen-Programm eines früheren Übungszettels, bei dem zu einer Postleitzahl der Ort mittels einer Instanzmethode ermittelt wurde. Leiten Sie von der vorhandenen Klasse eine Unterklasse ab, welche die Ortsermittlung sowohl für deutsche als auch für Postleitzahlen im Vereinigten Königreich beherrscht (dann übergeben als String). Verwenden Sie die Datei uk-postcode-database-csv.csv. Der erste Eintrag einer Zeile ist die Postleitzahl, der vorletzte ist der Ort, alles andere können Sie ignorieren.