Objektorientierte Programmiertechniken
LVA 185.A01, VU, 3 Ects, 2012/2013 W

8. Übungsaufgabe

Themen:

dynamische Typinformation, homogene Übersetzung von Generizität, Annotationen

Termine:

Ausgabe:	05.12.2012
reguläre Abgabe:	12.12.2012, 12:00 Uhr
nachträgliche Abgabe:	19.12.2012, 12:00 Uhr

Abgabeverzeichnis:

Gruppe/Aufgabe8

Programmaufruf:

java Test

Grundlage:

Skriptum, Schwerpunkt auf den Abschnitten 3.3.1 und 3.3.2 sowie Folien vom 5. Dezember

Aufgabe

Welche Aufgabe zu lösen ist:

Die landwirtschaftliche Produktion auf Bauernhöfen erfolgt heute unter Verwendung von Maschinen wie z.B. Traktoren. Jeder Bauernhof hat einen eindeutigen, unveränderlichen Namen und besitzt eine Menge von Traktoren. Jeder Traktor hat eine eindeutige, unveränderliche Nummer. Für jeden Traktor wird gespeichert, seit wievielen Stunden der Traktor bereits in Betrieb ist. Es gibt zwei Arten von Traktoren mit unterschiedlichen Motoren. Bei einem Traktor mit Dieselmotor wird gespeichert, wieviel Liter Diesel er seit Betriebsbeginn verbraucht hat (als ganze Zahl). Bei einem Traktor mit Biogasmotor wird gespeichert, wieviele Kubikmeter Gas er seit Betriebsbeginn

verbraucht hat (Gleitkommazahl). Jeder Traktor kann für seinen Einsatzzweck umgerüstet werden und entweder zum Säen mittels Drillmaschine oder zum Düngen mittels Düngerstreuer eingesetzt werden. Von jeder Drillmaschine ist die Anzahl der Säschare (ganze Zahl) bekannt, von jedem Düngerstreuer die Fassungskapazität seines Behälters in Liter (Gleitkommazahl). Zu jedem Zeitpunkt wird ein Traktor höchstens für eine Art von Aufgabe eingesetzt.

Entwickeln Sie Java-Klassen bzw. Interfaces zur Darstellung von Traktoren auch für unterschiedliche Einsatzarten. Folgende Funktionalität soll unterstützt werden:

- Erzeugen eines Traktors.
- Erhöhen der Betriebsstunden.
- Auslesen der Betriebsstunden.
- Erhöhen und Auslesen der verbrauchten Menge an Diesel eines Traktors mit Dieselmotor.
- Erhöhen und Auslesen der verbrauchten Menge an Gas eines Traktors mit Biogasmotor.
- Ändern der Einsatzart eines Traktors, wobei Informationen über frühere Einsatzarten dieses Traktors verloren gehen.
- Auslesen der Anzahl der Säschare oder der Fassungskapazität des Düngerbehälters.

Schreiben Sie eine Klasse *Bauernhof*, die Informationen über einen Bauernhof verwaltet und statistische Auswertungen über diesen Bauernhof ermöglicht. Jeder Bauernhof hat einen unveränderlichen Namen. Folgende Methoden sollen unterstützt werden:

- Erzeugen eines Bauernhofes.
- Einfügen von Traktoren in einen Bauernhof.
- Entfernen von Traktoren aus einem Bauernhof.
- Ändern der Informationen über Traktoren wie oben beschrieben.
- Methoden zum Berechnen folgender statistischer Werte:
    - Die durchschnittliche Anzahl der Betriebsstunden aller Traktoren eines Bauernhofs – alle Traktoren zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (Säen oder Düngen).
    - Die durchschnittliche Anzahl der Betriebsstunden aller Traktoren eines Bauernhofs aufgeschlüsselt nach der Art des Traktors (Dieseltraktor oder Biogastraktor).
    - Der durchschnittliche Dieselverbrauch aller Diesetraktoren eines Bauernhofs – alle zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (Säen oder Düngen).
    - Der durchschnittliche Gasverbrauch aller Biogastraktoren eines Bauernhofs – alle zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (Säen oder Düngen).
    - Die minimale und maximale Anzahl an Säscharen insgesamt und aufgeschlüsselt nach Art des Traktors (Dieseltraktor oder

Biogastraktor).
    - Die durchschnittliche Fassungskapazität des Düngerbehälters aller Traktoren insgesamt und aufgeschlüsselt nach Art des Traktors (Dieseltraktor oder Biogastraktor).

Zusätzlich soll der von Ihnen geschriebene Programmcode Meta-Informationen über die Klassen und Methoden enthalten. Versehen Sie jede selbst geschriebene Klasse und jede darin enthaltene Methode mit einer Annotation, die in einem String angibt, welches Gruppenmitglied hauptsächlich für die Entwicklung dieser Klasse bzw. Methode verantwortlich ist.

Die Klasse *Test* soll die wichtigsten Normal- und Grenzfälle (nicht interaktiv) überprüfen und die Ergebnisse in allgemein verständlicher Form in der Standardausgabe darstellen. Machen Sie unter anderem Folgendes:

- Erstellen Sie eine Menge von Bauernhöfen mit jeweils einigen Traktoren – wirklich eine *Menge* von Bauernhöfen (eine Form von Collection), nicht nur eine Ansammlung einzelner Variablen. Jeder Bauernhof in der Menge soll über seinen eindeutigen Namen angesprochen werden, und jeder Traktor eines Bauernhofes über seine eindeutige Nummer.
- Fügen Sie zu einigen Bauernhöfen einzelne Traktoren hinzu, entfernen Sie einzelne Traktoren, und ändern Sie die Informationen zu einzelnen Traktoren, wobei Sie Traktoren und Bauernhöfe nur über deren Nummern und Namen ansprechen.
- Berechnen Sie die statistischen Werte aller Bauernhöfe (wie oben beschrieben).
- Geben Sie die Namen der von Ihnen geschriebenen Klassen und Methoden zusammen mit den Personen, die dafür hauptsächlich verantwortlich sind, aus. Dabei sollen die Methodennamen sowie die verantwortlichen Personen ausschließlich über Reflection aus den Klassen mithilfe der Annotationen ermittelt werden.

Generizität, Arrays und vorgefertigte Container-Klassen dürfen zur Lösung dieser Aufgabe nicht verwendet werden. Ausgenommen davon sind nur Arrays als Ergebnis der Methode *getMethods* in *Class*, die zur Ermittlung der Methoden einer Klasse nötig ist. Vermeiden Sie mehrfach vorkommenden Code für gleiche oder ähnliche Programmteile.

Warum die Aufgabe diese Form hat:

Die gleichzeitige Unterscheidung von Diesetraktoren und Biogastraktoren sowie zwischen unterschiedlichen Einsatzarten stellt eine Schwierigkeit dar, für die es mehrere sinnvolle Lösungsansätze gibt. Sie werden irgendeine Form von Container selbst erstellen müssen, wobei die genaue Form und Funktionalität nicht vorgegeben ist. Da Container an mehreren Stellen benötigt werden, wäre die Verwendung von Generizität sinnvoll. Dadurch, dass Sie Generizität nicht verwenden dürfen und trotzdem mehrfache Vorkommen ähnlichen Codes

vermeiden sollen, werden Sie gezwungen, Techniken ähnlich denen einzusetzen, die der Compiler zur homogenen Übersetzung von Generizität verwendet. Vermutlich sind Typumwandlungen kaum vermeidbar. Sie sollen dadurch ein tieferes Verständnis des Zusammenhangs zwischen Generizität und Typumwandlungen bekommen. Daneben soll der Umgang mit Reflection und Annotationen in den wichtigsten Grundzügen geübt werden.

Was im Hinblick auf die Beurteilung zu beachten ist:

Der Schwerpunkt bei der Beurteilung liegt auf der vernünftigen Verwendung von dynamischer und statischer Typinformation. Kräftige Punkteabzüge gibt es für

- die Verwendung von Generizität bzw. von Arrays (außer als Ergebnis der Methode *getMethods*) oder vorgefertigten Container-Klassen
- mehrfach vorkommende gleiche oder ähnliche Programmteile (wenn vermeidbar)
- den unnötigen Verlust an statischer Typsicherheit
- Verletzungen des Ersetzbarkeitsprinzips bei Verwendung von Vererbungsbeziehungen (also Vererbungsbeziehungen, die keine Untertypbeziehungen sind)
- und mangelhafte Funktionalität des Programms.

Punkteabzüge gibt es unter anderem auch für mangelhafte Zusicherungen und falsche Sichtbarkeit.

Wie die Aufgabe zu lösen ist:

Es wird empfohlen, die Aufgabe zuerst mit Hilfe von Generizität zu lösen und in einem weiteren Schritt eine homogene Übersetzung der Generizität (wie im Skriptum beschrieben) händisch durchzuführen. Durch diese Vorgehensweise erreichen Sie eine statische Überprüfung der Korrektheit vieler Typumwandlungen und vermeiden den unnötigen Verlust an statischer Typsicherheit. Versehen Sie Ihre Klassen und Methoden von Anfang an mit den nötigen Annotationen, auch wenn Sie entsprechende Tests erst später hinzufügen.

Zur Lösung dieser Aufgabe ist die Verwendung von Typumwandlungen ausdrücklich erlaubt. Versuchen Sie trotzdem, die Anzahl der Typumwandlungen klein zu halten und so viel Typinformation wie möglich statisch vorzugeben. Das hilft Ihnen dabei, die Lösung überschaubar zu halten und einen unnötigen Verlust an statischer Typsicherheit zu vermeiden. Gehen Sie auch möglichst sparsam mit dynamischen Typabfragen und Ausnahmebehandlungen um.

Achten Sie darauf, dass Sie Divisionen durch 0 vermeiden. Führen Sie zumindest einen Testfall ein, bei dem eine statistische Auswertung ohne

entsprechende Vorkehrungen eine Exception aufgrund einer Division durch 0 auslösen würde.

Bedenken Sie, dass es mehrere sinnvolle Lösungsansätze für diese Aufgabe gibt. Wenn Sie einmal einen gangbaren Weg gefunden haben, bleiben Sie dabei, und vermeiden Sie es, zu viele Möglichkeiten auszuprobieren. Das könnte Ihnen viel Zeit kosten, ohne die Lösung zu verbessern.

Die Art der Verwendung eines Traktors für unterschiedene Einsatzzwecke kann sich im Laufe der Zeit ändern. Am besten stellt man solche Beziehungen über *Rollen* dar: Für jede Art von Traktor gibt es eine eigene Klasse mit den für die jeweilige Art typischen Daten, und ein gemeinsamer Obertyp ermöglicht den Zugriff auf diese Daten auf einheitliche Weise. In jedem Traktor gibt es einen Referenz auf die aktuelle Einsatzart (= die Rolle, die der Traktor gerade spielt). Wenn sich die Art ändert, braucht nur diese Referenz neu gesetzt zu werden. Durch geschickte Auswahl der Methoden einer Rolle sind die meisten Fallunterscheidungen vermeidbar, das heißt, Fallunterscheidungen werden durch dynamisches Binden ersetzt.

Was im Hinblick auf die Abgabe zu beachten ist:

Verzichten Sie auch bei der Lösung dieser Aufgabe auf die Verwendung von packages und Verzeichnissen innerhalb des Abgabeverzeichnisses *Gruppe/Aufgabe8*. Schreiben Sie (abgesehen von geschachtelten Klassen) nicht mehr als eine Klasse in jede Datei.