6. Programmieraufgabe

Objektorientierte Programmiertechniken

LVA-Nr. 185.A01
2021/2022 W
TU Wien

Themen:

dynamische Typinformation, homogene Übersetzung von Generizität

Ausgabe:

30. 11. 2021

Abgabe (Deadline):

7. 12. 2021, 12:00 Uhr

Abgabeverzeichnis:

`Aufgabe6`

Programmaufruf:

`java Test`

Grundlage:

Skriptum, Schwerpunkt auf 3.3.1 und 3.3.2

Kontext

Die Bäume im Wald haben sich gut entwickelt und haben eine beachtliche Größe erreicht. Jetzt ist es Zeit, einzelne Bäume zur Nutzung aus dem Wald zu entnehmen. Heutzutage werden auch bei nachhaltiger Waldnutzung die Bäume nicht mit der Hand gefällt und mit Pferden aus dem Wald gezogen. Die Forstbetriebe setzen dazu Holzvollernter (Harvester) ein. Holzvollernter sind riesige Holzerntemaschinen, die in einem Arbeitsgang einen Baum fällen, entasten und in Stücke schneiden können. Jeder Holzvollernter hat eine eindeutige, unveränderliche Nummer (ganze Zahl). Für jeden Holzvollernter wird gespeichert, seit wie vielen Stunden er bereits in Betrieb ist. Es gibt zwei Arten von Holzvollerntern mit unterschiedlichen Methoden zur Fortbewegung. Bei einem Holzvollernter mit Rädern wird die Wegstrecke in Metern gespeichert, die er seit Betriebsbeginn zurückgelegt hat (Gleitkommazahl). Bei einem Holzvollernter mit sechs Schreitbeinen (Schreiter) wird gespeichert, wieviele Schritte er seit Betriebsbeginn durchgeführt hat (als ganze Zahl). Der Arbeitskopf eines Holzvollernters kann je nach Einsatzzweck umgerüstet werden und entweder zum in Stücke Schneiden oder zum Erzeugen von Hackschnitzeln eingesetzt werden. Von jedem Arbeitskopf zum in Stücke Schneiden ist die maximal mögliche Länge eines Stücks in Meter (Gleitkommazahl) bekannt, von jedem Hackschnitzelkopf die maximale mögliche Dicke eines Baumes in Zentimeter (ganze Zahl). Zu jedem Zeitpunkt wird ein Holzvollernter höchstens für eine Art von Aufgabe eingesetzt.

Welche Aufgabe zu lösen ist

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

- Erzeugen eines Holzvollernters.
- Erhöhen der Betriebsstunden.
- Auslesen der Betriebsstunden.
- Erhöhen und Auslesen der Wegstrecke eines Holzvollernters mit Rädern.
- Erhöhen und Auslesen der Schritte eines schreitenden Holzvollernters.
- Ändern der Einsatzart eines Holzvollernters, wobei Informationen über frühere Einsatzarten dieses Holzvollernters verloren gehen.
- Auslesen der maximalen Stücklänge oder der maximalen Baumdicke.

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

- Erzeugen eines Forstbetriebs.
- Hinzufügen von Holzvollerntern zu einem Forstbetrieb.
- Entfernen von Holzvollerntern eines Forstbetriebs.
- Ändern der Informationen über Holzvollernter wie oben beschrieben.
- Methoden zum Berechnen folgender (statistischer) Werte:
  - Die durchschnittliche Anzahl der Betriebsstunden aller Holzvollernter eines Forstbetriebs – alle Holzvollernter zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (in Stücke schneiden oder Hackschnitzel erzeugen).
  - Die durchschnittliche Anzahl der Betriebsstunden aller Holzvollernter eines Forstbetriebs aufgeschlüsselt nach der Art des Holzvollernters (Radernter oder Schreiter).
  - Die durchschnittlich zurückgelegte Wegstrecke aller Radernter eines Forstbetriebs – alle zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (in Stücke schneiden oder Hackschnitzel erzeugen).
  - Die durchschnittliche Anzahl an Schritten aller Schreiter eines Forstbetriebs – alle zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (in Stücke schneiden oder Hackschnitzel erzeugen).
  - Die kleinste und größte maximale Stücklänge aller Holzvollernter mit Schneidearbeitskopf eines Forstbetriebs insgesamt und aufgeschlüsselt nach Art des Holzvollernters (Radernter oder Schreiter).
  - Die durchschnittliche Baumdicke aller Holzvollernter mit Hackschnitzelkopf eines Forstbetriebs insgesamt und aufgeschlüsselt nach Art des Holzvollernters (Radernter oder Schreiter).

Schreiben Sie eine Klasse `Region`, die Informationen über alle Forstbetriebe einer Region verwaltet. Jede Region hat einen unveränderlichen Namen. Folgende Methoden sollen unterstützt werden:

- Erzeugen einer Region (Objekt von `Region`).
- Hinzufügen von Forstbetrieben zu einer Region.
- Entfernen von Forstbetrieben einer Region.
- Anzeigen aller Forstbetriebe einer Region mit allen Informationen auf dem Bildschirm.

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 und ändern Sie mehrere Regionen mit mehreren Forstbetrieben mit jeweils einigen Holzvollerntern. Jeder Forstbetrieb einer Region soll über seinen eindeutigen Namen angesprochen werden, und jeder Holzvollernter eines Forstbetriebs über seine eindeutige Nummer.
- Fügen Sie zu Regionen einzelne Forstbetriebe hinzu, entfernen Sie einzelne Forstbetriebe, wobei Sie Forstbetriebe nur über deren Namen ansprechen.
- Fügen Sie zu einigen Forstbetrieben einzelne Holzvollernter hinzu, entfernen Sie einzelne Holzvollernter, und ändern Sie die Informationen zu einzelnen Holzvollerntern, wobei Sie Holzvollernter und Forstbetriebe nur über deren Nummern und Namen ansprechen.
- Berechnen Sie die statistischen Werte aller Forstbetriebe (wie oben beschrieben).

keine vorgefertigten Klassen

Code nicht duplizieren

Generizität, Arrays und vorgefertigte Container-Klassen dürfen zur Lösung dieser Aufgabe nicht verwendet werden. Vermeiden Sie mehrfach vorkommenden Code für gleiche oder ähnliche Programmteile.

Aufgabenaufteilung beschreiben

Daneben soll die Klasse `Test.java` als Kommentar eine kurze, aber verständliche Beschreibung der Aufteilung der Arbeiten auf die einzelnen Gruppenmitglieder enthalten – wer hat was gemacht.

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. 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 Sie viel Zeit kosten, ohne die Lösung zu verbessern.

Was im Hinblick auf die Beurteilung wichtig ist

Die insgesamt 100 für diese Aufgabe erreichbaren Punkte sind folgendermaßen auf die zu erreichenden Ziele aufgeteilt:

- Container richtig und wiederverwendbar implementiert, Typumwandlungen korrekt — 35 Punkte
- Geforderte Funktionalität vorhanden (so wie in Aufgabenstellung beschrieben) — 20 Punkte
- Lösung wie vorgeschrieben und sinnvoll getestet — 20 Punkte
- Zusicherungen richtig und sinnvoll eingesetzt — 15 Punkte
- Sichtbarkeit auf kleinstmögliche Bereiche beschränkt — 10 Punkte

Schwerpunkte berücksichtigen

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 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.

Aufgabe nicht abändern

Code nicht duplizieren

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

Warum die Aufgabe diese Form hat

Die gleichzeitige Unterscheidung von fahrenden und schreitenden Holzvollerntern 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.

Die Art der Verwendung eines Holzvollernters für unterschiedliche Einsatzzwecke kann sich im Laufe der Zeit ändern. Am besten stellt man

solche Beziehungen über Rollen dar: Für jede Art von Holzvollernter 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 Holzvollernter gibt es einen Referenz auf die aktuelle Einsatzart (= die Rolle, die der Holzvollernter 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

Schreiben Sie (abgesehen von geschachtelten Klassen) nicht mehr als eine Klasse in jede Datei. Verwenden Sie keine Umlaute in Dateinamen. Achten Sie darauf, dass Sie keine Java-Dateien abgeben, die nicht zu Ihrer Lösung gehören (alte Versionen, Reste aus früheren Versuchen, etc.).

keine Umlaute