6. Programmieraufgabe

Objektorientierte Programmiertechniken

LVA-Nr. 185.A01
2020/2021 W
TU Wien

Themen:

dynamische Typinformation, homogene Übersetzung von Generizität

Ausgabe:

18. 11. 2020

Abgabe (Deadline):

25. 11. 2020, 12:00 Uhr

Abgabeverzeichnis:

`Aufgabe6`

Programmaufruf:

`java Test`

Grundlage:

Skriptum, Schwerpunkt auf 3.3.1 und 3.3.2

Kontext

Auf Magerwiesen, die ein paar mal im Jahr gemäht werden, spielt sich das Leben nicht nur über der Erde sondern auch unter der Erde ab. Hamster und Ziesel haben hier ihre Baue und tragen durch ihre Grabtätigkeiten zur Bildung von Schwarzerde bei. Ziesel- und Hamsterbaue sind sehr ähnlich, doch gibt es kleine Unterschiede. So hat ein Hamsterbau zusätzlich eine Vorratskammer. Hamster sind überwiegend Einzelgänger, Ziesel bewohnen oft Bauten gemeinsam. Trotz der Unterschiede kommt es manchmal vor, dass ein leerer Zieselbau von einem Hamster oder ein leerer Hamsterbau von Zieseln übernommen wird. Bewohnte Zieselbauten kann man an der Oberfläche oft daran erkennen, dass um die Öffnungen herum Kotspuren zu finden sind. Sowohl Hamster als auch Ziesel stehen auf der roten Liste und es erfolgt ein regelmäßiges Monitoring der Hamster- und Zieselbaue einer Wiese. Dieses Beispiel hat nur in sehr eingeschränkter Weise mit der Realität zu tun. Falls Sie reale Hintergründe wissen wollen, finden Sie z.B. hier beim Naturschutzbund weiterführende Informationen.

Der Naturschutzbund benötigt ein einfaches Programm um die Baue dieser Nagetiere zu verwalten. Dazu werden verschiedene Informationen über Baue gespeichert. Ein Bau hat eine eindeutige Nummer (ganze Zahl) und enthält Bewohner (einen Hamster oder mehrere Ziesel), eine Wohnkammer und eine Menge von Röhren. Ist der Bewohner ein Hamster (es wird angenommen, dass nur ein Hamster den Bau bewohnt), wird das Volumen der Vorratskammer in Kubikzentimetern (Gleitkommazahl) gespeichert. Ist der Bewohner ein Ziesel, wird die Anzahl der Ziesel (ganze Zahl) gespeichert. Von der Wohnkammer wird das Volumen in Kubikzentimetern (Gleitkommazahl) gespeichert. Eine Röhre ist entweder eine Fallröhre oder eine geneigte Röhre. Jede Röhre hat einen eindeutigen Namen (Zeichenkette). Von einer Röhre wird der Durchmesser und die Länge in Zentimetern (ganze Zahl) gespeichert, von einer geneigten Röhre wird zusätzlich die Neigung in Grad (Gleitkommazahl) gespeichert. Da Röhren verschüttet oder neu gegraben werden können, kann sich die Anzahl der Röhren ändern. Auf einer Wiese befindet sich eine Menge unterschiedlicher Baue. Von einer Wiese wird eine eindeutige Nummer (ganze Zahl) und die Länge und Breite in Metern (ganze Zahl) gespeichert.

Welche Aufgabe zu lösen ist

Entwickeln Sie Java-Klassen bzw. Interfaces zur Darstellung von Bauen. Folgende Funktionalität soll unterstützt werden:

- Erzeugen einer Röhre mit einem eindeutigem Namen, einem Durchmesser, einer Länge und im Fall einer geneigten Röhre der Neigung.
- Erzeugen eines Baus mit eindeutiger Nummer und einem Bewohner.
- Auslesen der Nummer eines Baus.

- Auslesen der Anzahl der Röhren eines Baus.
- Auslesen des Volumens der Wohnkammer eines Baus.
- Auslesen des Volumens der Vorratskammer eines von einem Hamster bewohnten Baus.
- Auslesen der Anzahl der Bewohner eines von Ziesel bewohnten Baus.
- Ändern der Bewohner eines Baus.
- Hinzufügen einer Röhre zu einem Bau.
- Entfernen einer Röhre von einem Bau.
- Auslesen der Werte einer Röhre.

Schreiben Sie eine Klasse `Wiese`, die Informationen einer Wiese verwaltet und statistische Auswertungen über diese Wiese ermöglicht. Jede Wiese hat eine unveränderliche Kastralnummer (ganze Zahl). Folgende Methoden sollen unterstützt werden:

- Erzeugen einer Wiese mit eindeutiger Kastralnummer und einer Länge und Breite.
- Hinzufügen von Bauen zu einer Wiese.
- Entfernen von Bauen von einer Wiese.
- Ändern der Informationen von Bauen wie oben beschrieben.
- Methoden zum Berechnen folgender (statistischer) Werte:
  - Die Anzahl der Röhren eines Baus, die Anzahl der Baue einer Wiese, die Anzahl der Baue einer Wiese pro Hektar Wiesenfläche.
  - Die durchschnittliche Anzahl der Röhren eines Baus, einmal alle Baue zusammen und zusätzlich aufgeschlüsselt nach der Art der Bewohner (Hamster, Ziesel).
  - Der durchschnittliche Durchmesser der Röhren eines Baus, einmal alle Baue zusammen und zusätzlich aufgeschlüsselt nach der Art der Bewohner (Hamster, Ziesel).
  - Das durchschnittliche Volumen der Wohnkammer eines Baus, einmal alle Baue zusammen und zusätzlich aufgeschlüsselt nach der Art der Bewohner (Hamster, Ziesel).
  - Die durchschnittliche Länge aller Fallröhren.
  - Die durchschnittliche Neigung aller geneigten Röhren.
  - Das durchschnittliche Volumen der Vorratskammer eines von einem Hamster bewohnten Baus.
  - Die durchschnittliche Anzahl der Ziesel eines von Zieseln bewohnten Baus.

Schreiben Sie eine Klasse `Bundesland`, die Informationen über alle Wiesen eines Bundeslandes verwaltet. Jedes Bundesland hat einen unveränderlichen Namen. Folgende Methoden sollen unterstützt werden:

- Erzeugen eines Bundeslandes (Objekt von `Bundesland`).
- Hinzufügen einer Wiese zu einem Bundesland.
- Entfernen einer Wiese von einem Bundesland.
- Anzeigen aller Wiesen eines Bundeslandes 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 Bundesländer mit mehreren Wiesen mit jeweils einigen Bauen mit einigen Röhren. Jede Wiese eines Bundeslandes und jeder Bau einer Wiese soll über ihre (seine) eindeutige Nummer angesprochen werden, und jede Röhre eines Baus über ihren eindeutigen Namen.
- Fügen Sie einzelne Wiesen zu Bundesländern hinzu, entfernen Sie einzelne Wiesen, wobei Sie Wiesen nur über deren Nummern ansprechen.
- Fügen Sie einzelne Baue zu einigen Wiesen hinzu, entfernen Sie einzelne Baue, und ändern Sie die Informationen zu einzelnen Bauen, wobei Sie Wiesen und Baue nur über deren Nummern ansprechen.
- Fügen Sie zu einigen Bauen einzelne Röhren hinzu und entfernen Sie einzelne Röhren, wobei Sie Baue und Röhren nur über deren Namen oder deren Nummern ansprechen.
- Berechnen Sie die statistischen Werte aller Wiesen und des Bundeslandes (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:

Schwerpunkte berücksichtigen

- 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

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

Aufgabe nicht abändern

- die Verwendung von Generizität bzw. von Arrays oder vorgefertigten Container-Klassen

Code nicht duplizieren

- 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.

Warum die Aufgabe diese Form hat

Die gleichzeitige Unterscheidung von unterschiedlichen Röhren, Bauen, Wiesen und Bundesländern 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 Bewohner eines Baus können sich im Laufe der Zeit ändern. Am besten stellt man solche Beziehungen über Rollen dar: Für jede Art von Bewohnern 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 Bau gibt es eine Referenz auf die aktuellen Bewohner. Wenn sich die Bewohner ändern, 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