7. Programmieraufgabe

Objektorientierte Programmiertechniken

LVA-Nr. 185.A01
2017/2018 W
TU Wien

Themen:

kovariante Probleme, mehrfaches dynamisches Binden

Ausgabe:

6. 12. 2017

Abgabe (Deadline):

13. 12. 2017, 12:00 Uhr

Abgabeverzeichnis:

`Aufgabe7`

Programmaufruf:

`java Test`

Grundlage:

Skriptum, Schwerpunkt auf 3.3.3 und 3.4

Kontext

Installateurmeister Werner installiert nicht nur Heizungsanlagen, sondern auch Industrieanlagen. Daher hat er in seinem Lager Rohre mit verschiedenen Eigenschaften den Temperaturbereich und die Säurebeständigkeit betreffend:

- normales Rohr, kleiner Temperaturbereich
- normales Rohr, großer Temperaturbereich
- säurebeständiges Rohr, kleiner Temperaturbereich
- säurebeständiges Rohr, großer Temperaturbereich

Von jedem Rohr ist die Länge in cm (Gleitkommazahl) und der Preis in Cent (ganze Zahl) bekannt. Für Heizungssysteme werden nur normale Rohre verwendet, für Industrieanlagen nur säurebeständige. Rohre mit kleinem Temperaturbereich sind preiswerter als Rohre mit großem Temperaturbereich. Daher sollen für jede Anlage nur die exakt passenden Rohre verwendet werden, mit Ausnahme der folgenden Regel: falls kein passendes Rohr mit kleinem Temperaturbereich gelagert ist, darf auch ein passendes Rohr mit großem Temperaturbereich verwendet werden.

Welche Aufgabe zu lösen ist

Entwickeln Sie ein Programm zur Verwaltung der Rohre im Lager des Installateurs und der Rohre einer Anlage. Zumindest folgende Methoden sind zu entwickeln:

- `lagere` lagert ein Rohr im Lager des Installateurs ein.
- `montiere` gibt ein passendes Rohr für eine Anlage zurück und entfernt dieses aus dem Lager. Falls kein passendes Rohr existiert, wird `null` zurückgeliefert.
- `demontiere` entfernt ein Rohr aus einer Anlage und lagert dieses im Lager des Installateurs ein.
- `lagerwert` zeigt die Summe aller Preise der Rohre im Lager auf dem Bildschirm an.
- `anlagenwert` zeigt die Summe aller Preise der Rohre einer Anlage auf dem Bildschirm an.
- `lagerliste` zeigt alle Rohre des Lagerbestands mit allen Informationen auf dem Bildschirm an.
- `anlagenliste` zeigt alle Rohre einer Anlage mit allen Informationen auf dem Bildschirm an.

Die Klasse `Test` soll wie üblich die wichtigsten Normal- und Grenzfälle überprüfen und die Ergebnisse in allgemein verständlicher Form darstellen. Dabei sind Instanzen aller in der Lösung vorkommenden Typen zu erzeugen. Auch für das Lager des Installateurs und für Anlagen mit mehreren Rohren sind eigene Objekte zu erzeugen, und mindestens 3 unterschiedliche Anlagen sind zu testen. Testfälle sind so zu gestalten, dass sich deklarierte Typen von Variablen im Allgemeinen von den dynamischen Typen ihrer Werte unterscheiden.

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.

Aufgabenaufteilung beschreiben

In der Lösung der Aufgabe dürfen Sie folgende Sprachkonzepte *nicht* verwenden:

- dynamische Typabfragen `getClass` und `instanceof` sowie Typumwandlungen;
- bedingte Anweisungen wie `if`- und `switch`-Anweisungen sowie bedingte Ausdrücke (also Ausdrücke der Form `x?y:z`), die Typabfragen emulieren (d.h., zusätzliche Felder eines Objekts, die einen Typ simulieren und abfragen sind nicht erlaubt; z.B. ein `enum` der Rohrarten ist nicht sinnvoll, weil Abfragen darauf nicht erlaubt sind; bedingte Anweisungen, die einem anderen Zweck dienen, sind dagegen schon erlaubt);
- Werfen und Abfangen von Ausnahmen.

Bauen Sie Ihre Lösung stattdessen auf (mehrfaches) dynamisches Binden auf.

Was im Hinblick auf die Beurteilung wichtig ist

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

- (mehrfaches) dynamisches Binden richtig verwendet, sinnvolle minimale Typhierarchie, möglichst geringe Anzahl an Methoden und gute Wiederverwendung — 40 Punkte
- Lösung wie vorgeschrieben und sinnvoll getestet — 20 Punkte
- Geforderte Funktionalität vorhanden (so wie in Aufgabenstellung beschrieben) — 15 Punkte
- Zusicherungen richtig und sinnvoll eingesetzt — 15 Punkte
- Sichtbarkeit auf kleinstmögliche Bereiche beschränkt — 10 Punkte

Schwerpunkte bei der Beurteilung liegen auf der selbständigen Entwicklung geeigneter Untertypbeziehungen und dem Einsatz (mehrfachen) dynamischen Bindens. Kräftige Punkteabzüge gibt es für

- die Verwendung der verbotenen Sprachkonzepte,

- die Verwechslung von statischem und dynamischem Binden (insbesondere die Verwechslung überladener Methoden mit Multimethoden),
- Verletzungen des Ersetzbarkeitsprinzips (also Vererbungsbeziehungen, die keine Untertypbeziehungen sind)
- und nicht der Aufgabenstellung entsprechende oder falsche Funktionalität des Programms.

Punkteabzüge gibt es unter anderem auch für mangelhafte Zusicherungen, schlecht gewählte Sichtbarkeit und unzureichendes Testen (z.B. wenn grundlegende Funktionalität nicht überprüft wird).

Wie die Aufgabe zu lösen ist

Vermeiden Sie Typumwandlungen, dynamische Typabfragen und verbotene bedingte Anweisungen von Anfang an, da es schwierig ist, diese aus einem bestehenden Programm zu entfernen. Akzeptieren Sie in einem ersten Entwurf eher kovariante Eingangsparametertypen bzw. Multimethoden und lösen Sie diese dann so auf, dass Java damit umgehen kann (unbedingt vor der Abgabe, da sich sonst sehr schwere Fehler ergeben). Halten Sie die Anzahl der Klassen, Interfaces und Methoden möglichst klein und überschaubar. Durch die Aufgabenstellung ist eine große Anzahl an Klassen und Methoden ohnehin kaum vermeidbar, und durch weitere unnötige Strukturierung oder Funktionalität könnten Sie leicht den Überblick verlieren.

Es gibt mehrere sinnvolle Lösungsansätze. Bleiben Sie bei dem ersten von Ihnen gewählten sinnvollen Ansatz und probieren Sie nicht zu viele Ansätze aus, damit Ihnen nicht die Zeit davonläuft. Unterschiedliche sinnvolle Ansätze führen alle zu etwa demselben hohen Implementierungsaufwand.

Warum die Aufgabe diese Form hat

Die Aufgabe lässt Ihnen viel Entscheidungsspielraum. Es gibt zahlreiche sinnvolle Lösungsvarianten. Die Form der Aufgabe legt die Verwendung kovarianter Eingangsparametertypen nahe, die aber tatsächlich nicht unterstützt werden. Daher wird mehrfaches dynamisches Binden (durch simulierte Multi-Methoden bzw. das Visitor-Pattern) bei der Lösung hilfreich sein. Alternative Techniken, die auf Typumwandlungen und dynamischen Typabfragen beruhen, sind ausdrücklich verboten. Durch dieses Verbot wird die Notwendigkeit für dynamisches Binden noch verstärkt. Sie sollen sehen, wie viel mit dynamischem Binden möglich ist, aber auch, wo ein übermäßiger Einsatz zu Problemen führen kann.

Was im Hinblick auf die Abgabe zu beachten ist

Gerade für diese Aufgabe ist es besonders wichtig, dass Sie (abgesehen von geschachtelten Klassen) nicht mehr als eine Klasse in jede Datei geben und auf aussagekräftige Namen achten. Sonst ist es schwierig, sich einen Überblick über Ihre Klassen und Interfaces zu verschaffen. 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