# 6. Programmieraufgabe

**Objektorientierte Programmiertechniken**

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

**Themen:**

dynamische Typinformation, homogene Übersetzung von Generizität

**Ausgabe:**

29. 11. 2017

**Abgabe (Deadline):**

6. 12. 2017, 12:00 Uhr

**Abgabeverzeichnis:**

`Aufgabe6`

**Programmaufruf:**

`java Test`

**Grundlage:**

Skriptum, Schwerpunkt auf 3.3.1 und 3.3.2

## Kontext

Max hat in seinem Haus schon neue energiesparende dimmbare LED-Leuchtmittel in Verwendung, aber auch noch alte Glühbirnen. Nun interessiert ihn, wie hoch sein Energieverbrauch ist. Dazu benötigt er ein Programm um seine Lampen zu verwalten. Max hat zwei Arten von Lampen (mit gleichen Fassungen), einfache und dimmbare. Jede Lampe hat eine eindeutige Nummer (ganze Zahl). Bei beiden Arten ist bekannt, wie viele Stunden diese Lampen bereits eingeschaltet waren (ganze Zahl). Nur bei einfachen Lampen ist zusätzlich bekannt, welche Farbe die Lampe hat (Weiß, Rot oder Blau). Bei dimmbaren Lampen ist zusätzlich bekannt, zu wie viel Prozent die Lampe gedimmt ist (Gleitkommazahl). Bei Leuchtmitteln ist ihre Nennleistung in Watt (ganze Zahl) bekannt. Bei Glühbirnen ist zusätzlich die maximale Temperatur in Grad Celsius (ganze Zahl) bekannt, die die Glühbirne im Betrieb erreichen kann. Bei LED-Leuchtmitteln ist ebenso die Nennleistung, aber darüber hinaus auch der Lichtstrom in Lumen (Gleitkommazahl) bekannt.

## Welche Aufgabe zu lösen ist

Entwickeln Sie Java-Klassen bzw. Interfaces zur Darstellung von Lampen für unterschiedliche Leuchtmittel. Folgende Funktionalität soll unterstützt werden:

- Erzeugen einer Lampe mit einer eindeutigen Nummer.
- Auslesen der Nummer einer Lampe.
- Auslesen der Einschaltdauer einer Lampe.
- Ändern der Einschaltdauer einer Lampe.
- Auslesen des Dimmgrades einer dimmbaren Lampe.
- Ändern des Dimmgrades einer dimmbaren Lampe.
- Ändern des Leuchtmittels einer Lampe, wobei die Informationen über die Einschaltdauer des vorhergehenden Leuchtmittels verloren gehen.

Schreiben Sie eine Klasse `Zimmer`, die Informationen über ein Zimmer verwaltet und statistische Auswertungen über dieses Zimmer ermöglicht. Jedes Zimmer hat einen unveränderlichen Namen (Zeichenkette). Folgende Methoden sollen unterstützt werden:

- Erzeugen eines Zimmers.
- Hinzufügen von Lampen zu einem Zimmer.
- Entfernen von Lampen von einem Zimmer.

- Ändern der Informationen von Lampen wie oben beschrieben.
- Methoden zum Berechnen folgender statistischer Werte:
  - Die durchschnittliche Nennleistung einer Lampe in Watt aller Lampen eines Zimmers – alle Lampen zusammen und zusätzlich aufgeschlüsselt nach den Leuchtmittelarten (LED oder Glühbirne).
  - Die durchschnittliche Nennleistung einer Lampe in Watt aller Lampen eines Zimmers aufgeschlüsselt nach den Lampenarten (einfach oder gedimmt).
  - Die durchschnittliche Einschaltdauer aller einfachen Lampen eines Zimmers – alle zusammen und zusätzlich aufgeschlüsselt nach den Leuchtmittelarten (LED oder Glühbirne).
  - Die durchschnittliche Einschaltdauer aller dimmbaren Lampen eines Zimmers – alle zusammen und zusätzlich aufgeschlüsselt nach den Leuchtmittelarten (LED oder Glühbirne).
  - Die durchschnittliche Lichtausbeute aller Lampen eines Zimmers mit LED-Leuchtmittel in Lumen pro Watt.
  - Die maximale Temperatur aller Lampen eines Zimmers mit Glühbirnen in Grad Celcius.

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

- Erzeugen eines Hauses.
- Hinzufügen von Zimmern zu einem Haus.
- Entfernen von Zimmern aus einem Haus.
- Anzeigen aller Zimmer eines Hauses mit allen Informationen (inklusive Lampen) auf dem Bildschirm.
- Der Gesamtverbrauch aller Lampen eines Hauses über die gesamte Einschaltdauer in Kilowattstunden.

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 Häuser mit mehreren Zimmern mit jeweils einigen Lampen. Jedes Zimmer eines Hauses soll über seinen eindeutigen Namen angesprochen werden, und jede Lampe eines Zimmers über ihre eindeutige Nummer.
- Fügen Sie zu Häusern einzelne Zimmer hinzu, entfernen Sie einzelne Zimmer, wobei Sie Zimmer nur über deren Namen ansprechen.
- Fügen Sie zu einigen Zimmern einzelne Lampen hinzu, entfernen Sie einzelne Lampen, und ändern Sie die Informationen zu einzelnen Lampen, wobei Sie Lampen und Zimmer nur über deren Nummern und Namen ansprechen.

- Berechnen Sie die statistischen Werte aller Zimmer und des Hauses (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 Ihnen 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

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

Aufgabe nicht abändern

- mehrfach vorkommende gleiche oder ähnliche Programmteile (wenn vermeidbar)

Code nicht duplizieren

- 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 Lampen sowie zwischen unterschiedlichen Leuchtmitteln 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.

## 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