Objektorientierte Programmiertechniken
LVA 185.A01, VU, 3 ECTS, 2014/2015 W

# 8. Übungsaufgabe

**Themen:**

dynamische Typinformation, homogene Übersetzung von Generizität, Annotationen

**Termine:**

Ausgabe: 03.12.2014
Abgabe: 10.12.2014, 12:00 Uhr

**Abgabeverzeichnis:**

Gruppe/Aufgabe8

**Programmaufruf:**

java Test

**Grundlage:**

Skriptum, Schwerpunkt auf den Abschnitten 3.3.1 und 3.3.2 sowie 4.3

## Aufgabe

**Welche Aufgabe zu lösen ist:**

Nach den Erfolgen der Rosetta-Mission und einem erfolgreichen Start der Sonde Hayabusa-2 mit dem Erkundungsmodul MASCOT möchten die Raumfahrtagenturen mit weiteren Raumsonden unser Sonnensystem erkunden. Dazu werden die Raumsonden, die einen eindeutigen, unveränderlichen Namen haben, mit mehreren Erkundungsrobotern ausgestattet. Jeder Erkundungsroboter hat eine eindeutige, unveränderliche Nummer (ganze Zahl). Für jeden Erkundungsroboter wird gespeichert, seit wie vielen Stunden er bereits in Betrieb ist. Es gibt zwei Arten von Erkundungsrobotern mit unterschiedlichen Methoden zur Fortbewegung. Bei einem Erkundungsroboter mit Rädern wird die Wegstrecke in Metern gespeichert, die er seit Betriebsbeginn zurückgelegt hat (Gleitkommazahl). Bei einem

Erkundungsroboter, der mittels einer exzentrischen Schwungmasse Sprünge durchführt, wird gespeichert, wieviele Sprünge er seit Betriebsbeginn durchgeführt hat (als ganze Zahl). Jeder Erkundungsroboter kann für seinen Einsatzzweck umgerüstet werden und entweder zum Fotografieren mittels Kamera oder zum Bohren mittels Bohrer eingesetzt werden. Von jeder Kamera ist die Anzahl der Pixel des Sensors (ganze Zahl) bekannt, von jedem Bohrer die Länge seines Bohrers in Zentimeter (Gleitkommazahl). Zu jedem Zeitpunkt wird ein Erkundungsroboter höchstens für eine Art von Aufgabe eingesetzt.

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

- Erzeugen eines Erkundungsroboters.
- Erhöhen der Betriebsstunden.
- Auslesen der Betriebsstunden.
- Erhöhen und Auslesen der Wegstrecke eines Erkundungsroboters mit Rädern.
- Erhöhen und Auslesen der Sprunganzahl eines springenden Erkundungsroboters.
- Ändern der Einsatzart eines Erkundungsroboters, wobei Informationen über frühere Einsatzarten dieses Erkundungsroboters verloren gehen.
- Auslesen der Anzahl der Kamerapixel oder der Länge des Bohrers.

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

- Erzeugen einer Raumsonde.
- Hinzufügen von Erkundungsrobotern einer Raumsonde.
- Entfernen von Erkundungsrobotern einer Raumsonde.
- Ändern der Informationen über Erkundungsroboter wie oben beschrieben.
- Methoden zum Berechnen folgender statistischer Werte:
    - Die durchschnittliche Anzahl der Betriebsstunden aller Erkundungsroboter einer Raumsonde – alle Erkundungsroboter zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (Fotografieren oder Bohren).
    - Die durchschnittliche Anzahl der Betriebsstunden aller Erkundungsroboter einer Raumsonde aufgeschlüsselt nach der Art des Erkundungsroboters (Radroboter oder Springroboter).
    - Die durchschnittlich zurückgelegte Wegstrecke aller Radroboter einer Raumsonde – alle zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (Fotografieren oder Bohren).
    - Die durchschnittliche Anzahl an Sprüngen aller Springroboter einer Raumsonde – alle zusammen und zusätzlich aufgeschlüsselt nach den Einsatzarten (Fotografieren oder Bohren).

    - Die minimale und maximale Anzahl an Kamerapixeln aller Erkundungsroboter einer Raumsonde insgesamt und aufgeschlüsselt nach Art des Erkundungsroboters (Radroboter oder Springroboter).
    - Die durchschnittliche Länge des Bohrers aller Erkundungsroboter einer Raumsonde insgesamt und aufgeschlüsselt nach Art des Erkundungsroboters (Radroboter oder Springroboter).

Schreiben Sie eine Klasse *Mission*, die Informationen über eine Mission verwaltet. Jede Mission hat einen unveränderlichen Namen. Folgende Methoden sollen unterstützt werden:

- Erzeugen einer Mission.
- Hinzufügen von Raumsonden einer Mission.
- Entfernen von Raumsonden einer Mission.
- Anzeigen aller Raumsonden einer Mission auf dem Bildschirm.

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 und ändern Sie mehrere Missionen mit mehreren Raumsonden mit jeweils einigen Erkundungsrobotern. Jede Raumsonde einer Mission soll über ihren eindeutigen Namen angesprochen werden, und jeder Erkundungsroboter einer Raumsonde über seine eindeutige Nummer.
- Fügen Sie zu Missionen einzelne Raumsonden hinzu, entfernen Sie einzelne Raumsonden, wobei Sie Raumsonden nur über deren Namen ansprechen.
- Fügen Sie zu einigen Raumsonden einzelne Erkundungsroboter hinzu, entfernen Sie einzelne Erkundungsroboter, und ändern Sie die Informationen zu einzelnen Erkundungsrobotern, wobei Sie Erkundungsroboter und Raumsonden nur über deren Nummern und Namen ansprechen.
- Berechnen Sie die statistischen Werte aller Raumsonden (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 fahrenden und springenden Erkundungsrobotern 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:**

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

| | |
|---|---|
| Container richtig und wiederverwendbar implementiert, Typumwandlungen korrekt | 30 Punkte |
| Annotationen und Reflexion wie vorgeschrieben und sinnvoll eingesetzt | 20 Punkte |
| Geforderte Funktionalität vorhanden (so wie in Aufgabenstellung beschrieben) | 15 Punkte |
| Lösung wie vorgeschrieben und sinnvoll getestet | 15 Punkte |
| Zusicherungen richtig und sinnvoll eingesetzt | 10 Punkte |
| Sichtbarkeit auf so kleine Bereiche wie möglich 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 (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 Erkundungsroboters für unterschiedliche Einsatzzwecke kann sich im Laufe der Zeit ändern. Am besten stellt man solche Beziehungen über *Rollen* dar: Für jede Art von Erkundungsroboter 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 Erkundungsroboter gibt es einen Referenz auf die aktuelle Einsatzart (= die Rolle, die der Erkundungsroboter 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. Achten Sie darauf, dass Sie keine Java-Dateien abgeben, die nicht zu Ihrer Lösung gehören (alte Versionen, Reste aus früheren Versuchen, etc.).