Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Christian Böhm
Annahita Oswald, Bianca Wackersreuther

München, 22.01.2010

Einführung in die Programmierung

WS 2009/10

Übungsblatt 11: Interfaces, Ausnahmen

Besprechung: 01./03./04./05.02.2010

Hinweise:
Die Abgabe der Aufgaben auf diesem Blatt ist freiwillig. Die behandelten Inhalte sind Stoff der Nachholklausur. Um Ihre individuelle Lösung einschätzen zu können, bieten wir Ihnen auch weiterhin die Möglichkeit der Korrektur an. Weitere Bonuspunkte können nicht erzielt werden.

Aufgabe 11-1 *Throwable, Error, Exception*

Sie haben in der Vorlesung verschiedene Kategorien von Ausnahmen kennengelernt: `Error`, `Exception` und als besondere Unterklasse von `Exception` die `RuntimeException`. Alle diese Klassen von Ausnahmen sind Unterklassen von `Throwable`, d.h. sie können geworfen werden.

Während Exceptions im Allgemeinen behandelt oder (explizit) weitergegeben werden müssen (deswegen nennt man sie auch *checked Exceptions*), gilt dies für Objekte der Klasse `RuntimeException` und ihrer Unterklassen nicht (diese heißen daher auch *unchecked Exceptions*). Das Auftreten von *checked Exceptions* liegt normalerweise außerhalb des vom Programm kontrollierbaren Bereiches, während auftretende *unchecked Exceptions* normalerweise auf Fehler in der Programmlogik hinweisen.

Betrachten Sie nun folgende fünf Situationen. Jede dieser Situationen hat zur Folge, dass vom Java-Laufzeitsystem ein Objekt der Klasse `Throwable` geworfen wird. Geben Sie jeweils an, zu welcher der Kategorien *Error*, *unchecked Exception* oder *checked Exception* dieses Objekt gehört. Beantworten Sie außerdem jeweils die Fragen: Darf das Java-Programm explizit angeben, was damit geschehen soll? Muss es das explizit angeben? Sollte es das normalerweise explizit angeben, wenn es darf, aber nicht muss?

1. Ein Array `a` wurde ordnungsgemäß erzeugt, danach ist der Ausdruck `a[n]` erlaubt, aber der Wert von `n` ist zum Ausführungszeitpunkt negativ (`ArrayIndexOutOfBounds`).

2. Ein neues Objekt soll erzeugt werden, aber dafür ist kein Speicherplatz mehr verfügbar, in dem es gespeichert werden kann (`OutOfMemory`).

3. Eine Datei wurde zum Lesen geöffnet und daraufhin überprüft, dass das Dateiende noch nicht erreicht ist. Unmittelbar vor der nächsten Leseoperation wird die Stromversorgung des Geräts unterbrochen, auf dem die Datei gespeichert ist. Die nächste Leseoperation kann damit nichts mehr von der Datei lesen, versucht also rein physikalisch (wenn auch nicht logisch) über das Ende der Datei hinaus zu lesen (`EOF`).

4. Die Klasse `Main` benutzt eine Methode `m()` aus einer Klasse `K`. Nachdem beide Dateien fehlerlos übersetzt wurden, wird die Datei `K.java` editiert, die Methode `m()` umbenannt zu `n()`, dann die Datei `K.java` neu übersetzt. Danach wird das unveränderte Hauptprogramm wieder gestartet, aber die von ihm verwendete Methode `m()` existiert in der Klasse `K` gar nicht mehr (`NoSuchMethod`, `IncompatibleClassChange`).

5. Eine Variable `v` ist vom Typ `K`, der eine Unterklasse von `Object` ist, also ist der Ausdruck `v.toString()` erlaubt, aber zum Ausführungszeitpunkt hat die Variable `v` den Wert **`null`** (`NullPointer`).

Aufgabe 11-2 *Interfaces und Ausnahmen*

Eine Matrix ist ein rechteckiges Schema von Zahlen oder anderen Größen, die addiert und multipliziert werden können. Viele von Ihnen kennen das Rechnen mit Matrizen aus der linearen Algebra, für die anderen weisen wir kurz auf die Eigenschaften hin, die wir hier benötigen.

Wenn Matrizen von der Größe her zusammenpassen, ist es möglich, sie zu addieren oder miteinander zu multiplizieren. Das folgende Beispiel

$$A := \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{pmatrix}$$

zeigt eine Matrix A mit drei Zeilen und vier Spalten. Allgemein spricht man von einer $m \times n$-Matrix mit m Zeilen und n Spalten. Die Werte a_{ij} entsprechen dem Wert der Matrix in der i-ten Zeile und j-ten Spalte.
Hier benötigen wir folgende Rechenvorschriften:
Die Summe zweier $m \times n$-Matrizen A und B wird berechnet durch stellenweise Addition der Einträge der beiden Matrizen:

$$A + B := (a_{ij} + b_{ij})_{i=1,\ldots,m;j=1,\ldots,n}$$

Beispiel:

$$\begin{pmatrix} 2 & 5 & 4 \\ 1 & 2 & 3 \end{pmatrix} + \begin{pmatrix} 6 & 2 & 0 \\ 1 & 3 & 4 \end{pmatrix} = \begin{pmatrix} 2+6 & 5+2 & 4+0 \\ 1+1 & 2+3 & 3+4 \end{pmatrix} = \begin{pmatrix} 8 & 7 & 4 \\ 2 & 5 & 7 \end{pmatrix}$$

Es können nur Matrizen mit der gleichen Anzahl an Zeilen und der gleichen Anzahl an Spalten addiert werden.
Eine Matrix A wird mit einem Skalar λ multipliziert, indem alle Einträge der Matrix mit dem Skalar multipliziert werden:

$$\lambda * A = (\lambda * a_{ij})_{i=1,\ldots,m;j=1,\ldots,n}$$

Beispiel:

$$3 * \begin{pmatrix} 2 & 5 & 4 \\ 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 3*2 & 3*5 & 3*4 \\ 3*1 & 3*2 & 3*3 \end{pmatrix} = \begin{pmatrix} 6 & 15 & 12 \\ 3 & 6 & 9 \end{pmatrix}$$

Zwei Matrizen $A = (a_{ij})_{i=1,\ldots,m;j=1,\ldots,n}$ und $B = (b_{ij})_{i=1,\ldots,n;j=1,\ldots,o}$ werden folgendermaßen miteinander multipliziert:

$$A * B := \left(\sum_{k=1}^{n} a_{ik} * b_{kj} \right)_{i=1,\ldots,m;j=1,\ldots,o}$$

Das Ergebnis der Multiplikation einer $m \times n$ Matrix mit einer $n \times o$ Matrix ist also eine $m \times o$ Matrix.
Beispiel:

$$\begin{pmatrix} 2 & 5 & 4 \\ 1 & 2 & 3 \end{pmatrix} * \begin{pmatrix} 2 & 5 \\ 1 & 3 \\ 0 & 4 \end{pmatrix} = \begin{pmatrix} 2*2+5*1+4*0 & 2*5+5*3+4*4 \\ 1*2+2*1+3*0 & 1*5+2*3+3*4 \end{pmatrix} = \begin{pmatrix} 9 & 41 \\ 4 & 23 \end{pmatrix}$$

Ganz entsprechende Operationen sind Addition und Multiplikation einer Matrix A mit einem Vektor V bzw. umgekehrt, indem ein Vektor als Matrix mit einer Spalte (bzw. einer Zeile) aufgefasst wird.
Insbesondere kann deshalb die Multiplikation einer $m \times n$ Matrix A mit einem $n \times 1$ Vektor v als Transformation des Vektors in einen anderen Vektorraum aufgefasst werden, da das Ergebnis von $A * v$ ein $m \times 1$ Vektor ist.
Zwei Vektoren v und w gleicher Länge n können durch Transponieren eines der beiden Vektoren multipliziert werden. Fassen wir v als $n \times 1$, w als $1 \times n$ Vektor auf, so wird die Multiplikation $v * w$ *Tensorprodukt* genannt und das Ergebnis ist eine $n \times n$ Matrix. Fassen wir umgekehrt v als $1 \times n$, w als $n \times 1$ Vektor auf, so wird die Multiplikation $v * w$ *kanonisches Skalarprodukt* genannt, das Ergebnis ist eine Zahl.

In dieser Aufgabe sollen zwei Klassen `ReellerVektor` und `ReelleMatrix` zur Darstellung reellwertiger Vektoren und Matrizen programmiert werden. Implementieren Sie dazu die Interfaces `Vektor.java` und `Matrix.java`, die Sie auf der Vorlesungshomepage finden.

Achten Sie darauf, dass Sie auch geeignete Konstruktoren für beide Klassen zur Verfügung stellen, und verwenden Sie in sinnvoller Weise Exceptions, um ungültige Methodenaufrufe zu behandeln.

Hinweis: Die Klasse **java.text.NumberFormat** könnte hilfreich sein.