II.4.4 Exceptions

Mittwoch, 28. November 2018 09:30

- Exceptions treten auf, wenn semantische Regel während der Ausführung des Programms verletzt wird.
- Explizites Abfragen aller möglichen Fehlerquellen ist ggf. unübersichtlich.
- Laufzeitsystem v. Java meldet auftretende Exceptions automatisch.
- Auftretende ("geworfene") Exceptions können vom Programm selbst wieder behandelt werden ("können gefangen werden").
  Ansonsten bricht das Prog. mit Fehlermeldung ab.

Behandlung von Exceptions durch try-catch:

- Für jeden zu fangenden Exception-Typ gibt es einen eigenen Catch-Block.
- finally-Block kann fehlen. Sonst wird er am Ende auf jeden Fall ausgeführt (unabhängig davon, ob Exc. im Normalblock auftritt und unabh. davon, ob Exc. gefangen wird).

Klassenstruktur

- Throwable(String m) erzeugt Objekt mit Fehlermeldung m. Diese Fehlermeldung wird von toString()

bzw. getMessage() geliefert.
printStackTrace() gibt Information über Laufzeitkeller zum Zeitpunkt der Erzeugung des Ausnahmeobjekts

- Ausnahmeobjekte werden in Ausnahme-Situationen automatisch erzeugt. Sie können aber auch explizit durch Konstruktor erzeugt werden.
- unchecked exceptions:
  Error + RuntimeException
  müssen nicht gefangen werden.
  * Error: gravierende Fehler, von denen sich Prog. meist nicht "erholen" kann
  * RuntimeExceptions: sehr häufig auftretende

Exc. können (aber müssen nicht) gefangen werden.

Arbeit des try-catch-Blocks:

- A und B sind keine U-Klassen voneinander
- Wenn Exception in der Methode, in der sie auftritt, nicht gefangen wird, dann wird Methode abgebrochen und an die Stelle gesprungen, wo Methode aufgerufen. Nun wird versucht, die Ausnahme dort zu behandeln.
- Wenn eine Methode eine Exc. vom Typ A werfen könnte (und sie nicht

selbst fängt), dann muss der Methodenkopf "throws A" enthalten (es sei denn, A ist eine unchecked exception).

Bsp: Definiere eigene Exception-Klasse. Ist checked exception, d.h., sie muss gefangen werden. ⇒ Jede Methode, die fak aufruft, muss NNE fangen (oder selbst wieder "throws NNE" im Kopf haben).

Im Bsp:

bei x=3: Fakultaet von 3 ist 6

bei x=-3: Fehler! -3 < 0.

Exceptions haben Klassen-

hierarchie ⇒ mehrere
Catch-Blöcke zu einem
try-Block möglich.

Wenn catch-Blöcke von oben
nach unten abgearbeitet
werden und erst allgemeine
Exceptions gefangen werden,
dann würde man spätere
catch-Blöcke für speziellere
Exception-Typen nie erreichen.
⇒ Java verlangt, dass
catch-Blöcke für spezielle
Exception-Typen zuerst
kommen.

Bsp:
bei x=13: Fehler! Es trat die
folgende Ausnahme auf:

13 ist zu groß

↑

bei entspr. toString() - Methode

in TooBigNumberException

Bsp zur Demonstration von "finally":

bei x=3: Fakultaet von 3 ist 6.

Ende des try-catch Blocks

Ende der Methode test.

bei x=-3: Fehler! -3 < 0.

Ende des try-catch Blocks

Ende der Methode test.

bei x=13: Ende des try-catch Blocks

Fehler! Es trat die folgende Ausnahme auf:

Eingegebene Zahl 13 ist zu groß.

Verwende Ausnahmen Sinnvoll

- nicht als Ersatz für Sprünge
- nicht als Ersatz für Fallunterscheidungen (if, switch)