Informatik B - Objektorientierte Programmierung in Java

Vorlesung 06:

Ausnahme- und Fehlerbehandlung in Java

Inhalt

- Grundlagen der Fehlerbehandlung
- Behandlung von Ausnahmen/Exceptions
- Fehlerklassen
- Weitergabe von Exceptions
- Auslösen von Exceptions

Grundlagen der Fehlerbehandlung

- Wenn Programme nicht auf Ausnahmesituationen reagieren können, so führt das zu Abstürzen.
- Bei komplexen Programmsystemen und verteilten Anwendungen sind Programme ohne jede Reaktion auf Ausnahmesituationen nicht akzeptabel.
- Die Behandlung von Ausnahmesituationen mit Sprachkonstrukten, wie z.B. **`if`**, führt zu Unübersichtlichkeit und Verquickung mit dem normalen Programmcode.

Grundlagen der Fehlerbehandlung

- Java besitzt einen Mechanismus zur strukturierten Behandlung von Fehlern, die während der Programmausführung auftreten.
- Grundkonzept dieses Mechanismus sind sogenannte Ausnahmen (engl. Exceptions).
- Java stellt Sprachmittel zur Verfügung, die eine systematische Behandlung solcher Ausnahmen ermöglicht.
- Es existiert eine klare Trennung zwischen funktionalen Programmteilen und Programmteilen zur Fehlerbehandlung.

Grundlagen der Fehlerbehandlung

- Grundprinzip des Exception-Mechanismus
  - Ein Laufzeitfehler oder eine vom Entwickler gewollte Bedingung löst eine Exception aus (engl. throwing).
  - Diese kann nun entweder von dem Programmteil, in dem sie ausgelöst wurde, behandelt werden (engl. catching), oder sie kann weitergegeben werden.
  - Wird die Exception weitergegeben, so hat der Empfänger der Exception erneut die Möglichkeit, sie entweder zu behandeln oder selbst weiterzugeben.
  - Wird die Exception von keinem Programmteil behandelt, so führt sie zum Abbruch des Programms und zur Ausgabe einer Fehlermeldung.

Behandlung von Exceptions

- Syntax

```
try {
        // Folge von Anweisungen
}
catch (Exception1 e) {
        // Reaktion auf Exception vom Typ Exception1
}
catch (Exception2 e) {
        // Reaktion auf Exception vom Typ Exception2
}
...
finally {
        // Abschließende Anweisungen
}
```


Behandlung von Exceptions

- Ablauf (**`try`**/**`catch`**)
  - Der **`try`**-Block enthält eine oder mehrere Anweisungen, bei deren Ausführung verschiedene Arten von Fehlern auftreten können.
  - Tritt ein Fehler auf, wird die normale Programmausführung unterbrochen, und der Programmablauf fährt mit der ersten Anweisung nach der ersten **`catch`**-Klausel fort, die den passenden Exceptiontyp deklariert hat.
  - Die **`catch`**-Klauseln werden in der Reihenfolge ihres Auftretens ausgewählt.

Behandlung von Exceptions

- Ablauf (**`finally`**)
  - Die **`finally`**-Klausel wird immer dann ausgeführt, wenn der zugehörige **`try`**-Block betreten wurde.
  - Die **`finally`**-Klausel wird insbesondere dann ausgeführt, wenn der **`try`**-Block durch eine der folgenden Anweisungen verlassen wurde:
    - wenn das normale Ende des **`try`**-Blocks erreicht wurde
    - wenn eine Exception aufgetreten ist, die durch eine **`catch`**-Klausel behandelt wurde
    - wenn eine Exception aufgetreten ist, die nicht durch eine **`catch`**-Klausel behandelt wurde
    - wenn der **`try`**-Block durch eine der Sprunganweisungen **`break`**, **`continue`** oder **`return`** verlassen werden soll

Behandlung von Exceptions

- Wird in Java eine Exception ausgelöst, dann wird die weitere Ausführung der Operation abgebrochen. Mit Hilfe des **`finally`**-Konstrukts können auch dann noch Anweisungen ausgeführt werden, wenn eine Exception eintritt.
- Der **`finally`**-Block wird ausgeführt, egal, was im **`try`**-Block passiert.
  - Einzige Ausnahme: Der **`try`**-Block wird mit exit() verlassen.
- Die **`finally`**-Klausel ist also der ideale Ort, um Aufräumarbeiten (Schließen von Dateien etc.) durchzuführen.

Behandlung von Exceptions

- Fehlerobjekte
  - In der **`catch`**-Klausel wird nicht nur die Art des abzufangenden Fehlers definiert, sondern auch ein formaler Parameter **`e`** angegeben, der beim Auftreten der Exception ein Fehlerobjekt übernehmen soll.
  - Fehlerobjekte sind dabei Objekte der Klasse **`Throwable`** oder einer ihrer Unterklassen.
  - Ein Fehlerobjekt wird vom Auslöser der Exception erzeugt und als Parameter an die **`catch`**-Klausel übergeben.
  - Ein Fehlerobjekt enthält Informationen über die Art des aufgetretenen Fehlers.

Behandlung von Exceptions

- Alle Laufzeitfehler in Java sind Unterklassen der Klasse **`Throwable`**.
- **`Throwable`** ist eine allgemeine Fehlerklasse, die im wesentlichen eine Klartext-Meldung speichern und einen Auszug des Laufzeit-Kellers ausgeben kann (der Laufzeit-Keller enthält während des Ablaufs eines Java-Programms alle aufgerufenen Operationen, die noch nicht abgeschlossen sind).

Behandlung von Exceptions

- Wichtige Operationen der Klasse **`Throwable`**
  - **`public String getMessage()`**
    liefert einen Fehlertext
  - **`public void printStackTrace()`**
    druckt einen Auszug aus dem Laufzeit-Keller auf **`System.err`**
  - **`public String toString()`**
    liefert den Namen der Fehlerklasse und einen Fehlertext

Behandlung von Exceptions

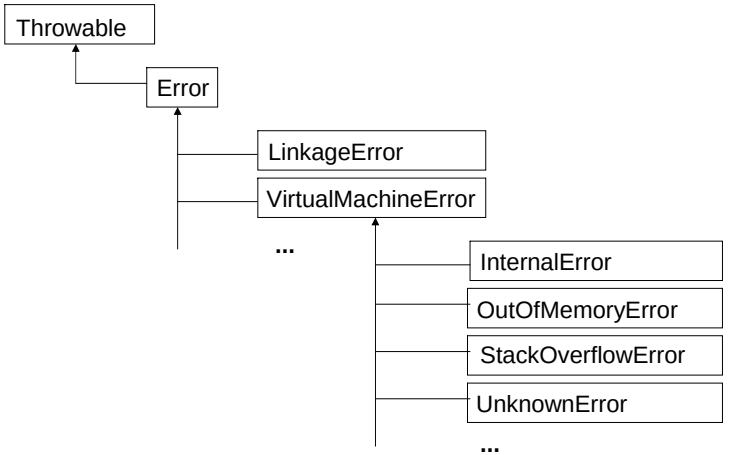
- Beispielprogramme:
  - Laufzeitfehler.java
    - Ohne Exceptionbehandlung: **`ArithmeticException`** bei Division durch Null führt zum Abbruch des Programms.
  - TryCatch.java
    - **`catch(ArithmeticException e)`** fängt diese ab und gibt Fehlertext und Laufzeitkeller aus.
  - TryCatchFinally.java
    - Geschachtelte **`try-catch`** Blöcke jeweils mit **`finally.`**
  - TryManyCatch.java
    - Mehrere **`catch`**-Klauseln, die in Reihenfolge ihres Auftretens ausgewählt werden.

Fehlerklassen

- Unterhalb von **`Throwable`** befinden sich zwei große Vererbungshierarchien für Fehlerklassen:
  - **`Error`**
    - Oberklasse aller schwerwiegenden Fehler
    - Diese werden hauptsächlich durch Probleme im Java-Interpreter (Java Virtual Machine) ausgelöst.
  - **`Exception`**
    - Oberklasse aller Fehler, die möglicherweise für das Programm selbst von Interesse sind
    - Fehler dieser Art signalisieren einen abnormen Zustand, der vom Programm abgefangen und behandelt werden kann.

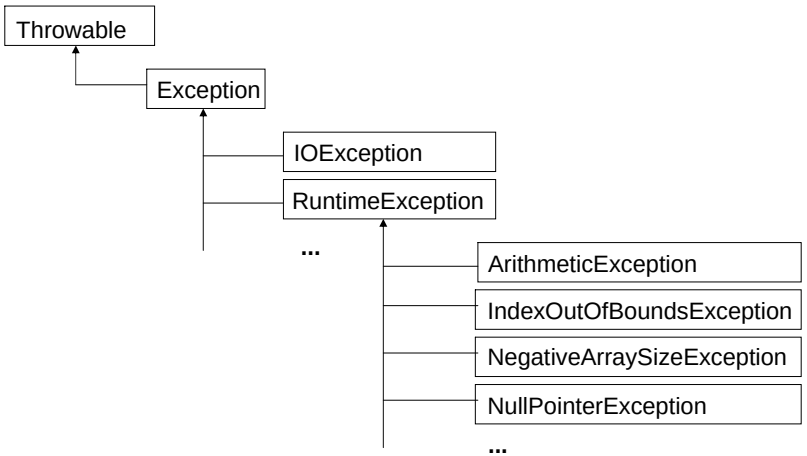
Fehlerklassen

- Schwerwiegende Fehler (Ausschnitt)



Fehlerklassen

- Programmausnahmen (Ausschnitt)



Weitergabe von Exceptions

- Catch-or-throw-Regel:

  Jede Exception muss behandelt oder weitergegeben werden.
- Jede mögliche Exception, die nicht behandelt, sondern weitergeben wird, ist mit Hilfe der **`throws`**-Klausel zu deklarieren.
- Dazu wird an das Ende der Methodensignatur das Schlüsselwort **`throws`** mit einer Liste aller Exceptions, die nicht behandelt werden sollen, angehängt.

Weitergabe von Exceptions

- Um den Aufwand durch die explizite Fehlerbehandlung bei der Entwicklung von Java-Programmen nicht zu groß werden zu lassen, gibt es zwei Exceptions von der Catch-or-throw-Regel:
  - Direkt unterhalb der Klasse **`Exception`** gibt es die Klasse **`RuntimeException`**, welche die Oberklasse aller Laufzeitfehler ist, die zwar deklariert werden können, aber nicht müssen.
  - Der Programmierer kann in diesem Fall entscheiden, ob er den entsprechenden Fehler deklarieren will oder nicht.
  - Ebenso verhält es sich mit der Klasse **`Error`**.

Weitergabe von Exceptions

- Beispiel für **`throws`**-Klausel

```
int liesGanzeZahl(BufferedReader din)
throws IOException
{
  int zahl;

  zahl = Integer.parseInt(din.readLine());
  return zahl;
}
```


Weitergabe von Exceptions

- Ablauf
  - Tritt eine Exception ein, sucht der Java-Interpreter zunächst nach einer die Anweisung unmittelbar umgebenden **`try-catch`**-Anweisung, die diesen Fehler behandelt.
  - Findet es eine solche nicht, wiederholt es die Suche sukzessive in allen umgebenden Blöcken.
  - Ist auch das erfolglos, wird der Fehler an den Aufrufer der Operation weitergebeben.
  - Dort beginnt die Suche von neuem, und der Fehler wird an die umgebenden Blöcke und schließlich an den Aufrufer weitergereicht.

Weitergabe von Exceptions

- Enthält auch die Operation **main** keinen Code, um den Fehler zu behandeln, bricht das Programm mit einer Fehlermeldung ab.
- Die **throw**-Anweisung hat den Charakter einer Sprunganweisung. Sie unterbricht das Programm an der aktuellen Stelle und verzweigt unmittelbar zu dem **catch** der umgebenden **try**-Klausel. Gibt es eine solche nicht, wird der Fehler an den Aufrufer weitergegeben. Tritt der Fehler innerhalb einer **try-catch**-Anweisung mit einer **finally**-Klausel auf, wird diese noch vor der Weitergabe des Fehlers ausgeführt.

Auslösen von Exceptions

- Exceptions in Java sind Objekte bestimmter Klassen und können somit behandelt werden wie andere Objekte auch.
- Ein Fehlerobjekt wird mit dem **new**-Operator instantiiert, z.B.

```
ArithmeticException ausnahmeObj;
ausnahmeObj = new ArithmeticException(...);
```

- Mit Hilfe der **throw**-Anweisung kann ein solches Objekt dazu verwendet werden, eine Exception auszulösen:

```
throw ausnahmeObj;
```


Auslösen von Exceptions

- Beispiel
  - Ermittlung, ob **n** Primzahl; wenn **n<=0**: Exception

```
public boolean isPrim(int n) throws ArithmeticException
{
   if (n <= 0) {
   throw new ArithmeticException("isPrim: Parameter <=0");
   }

   if (n == 1)
       return false;

   for (int i = 2; i <= n/2; ++i) {
       if (n % i == 0)
              return false;
   }
   return true;
}
```


Auslösen von Exceptions

- Die **throw**-Anweisung kann nicht nur dazu verwendet werden, neue Fehler auszulösen.
- Sie kann ebenfalls eingesetzt werden, um innerhalb der **catch**-Klausel einer **try-catch**-Anweisung das übergebene Fehlerobjekt erneut zu senden.
- In diesem Fall wird nicht noch einmal dieselbe **catch**-Klausel ausgeführt, sondern der Fehler wird gemäß den oben genannten Regeln an den umgebenden Block bzw. den Aufrufer weitergegeben.

Auslösen von Exceptions

- Beispiel

```
try {
        // Folge von Anweisungen
        ...
}
catch (Exception1 e) {
        // Lokale Fehlerbehandlung
        ...
        // Weitergabe zur übergeordneten
        // Fehlerbehandlung
        throw e;
}
finally {
        // Abschließende Anweisungen
        ...
}
```


Auslösen von Exceptions

- Selbstdefinierte Fehlerklassen
  - Aus den vorhanden Fehlerklassen der Java-Klassenbibliothek können eigene Fehlerklassen abgeleitet werden.
  - Diese sollten aus der Klasse **`Exception`** oder einer ihrer Unterklassen abgeleitet werden, nicht jedoch aus der Klasse **`Error`**.
  - Auch selbstdefinierte Exceptions müssen sich an die Catch-or-throw-Regel halten.

Auslösen von Exceptions

- Selbstdefinierte Fehlerklassen
  - Beispiel: Datum.java
    - Definiert:
      `IllegalesDatumException extends Exception`