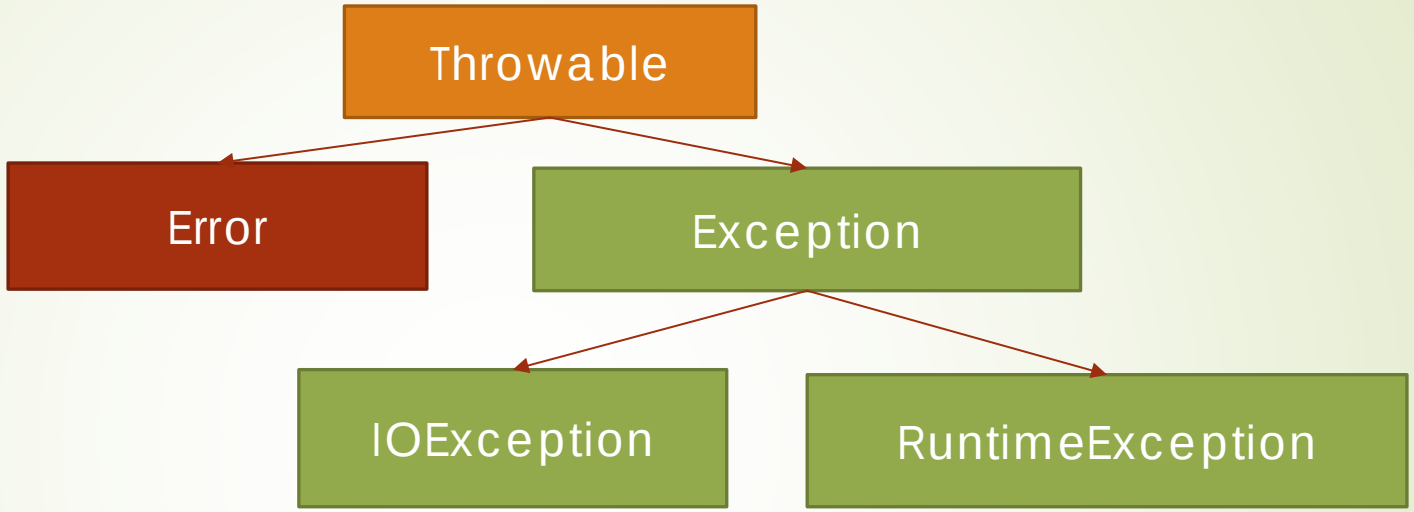
Java für Fortgeschrittene

Teil 5

Ausnahme Behandlung (Exceptiones)



- Exceptiones dienen zum Abfangen von Fehlern die durch das eigen Programm erzeugt wurden. Dies kann z.B. ein Laufzeitfehler sein oder ein IO-Fehler. Die Basisklasse lautet Exception. Der Fehler kann abgefangen werden und bearbeitet werden.
- Errors sind Fehler die von der JVM erzeugt werden. Da es sich auch bei Error um ein Thowable handelt lässt er sich zwar abfangen, er lässt sich jedoch nicht bearbeiten die Folge ist dass das Programm beendet werden muss bei solch einen Fehler.

Besondere Eigenschaften der Klasse „Exception/Throwable“

- Der Constructer kann kann entweder ohne Parameter aufgerufen werden oder ihm eine Message vom Typ String übergeben werden.
- Die Message kann auch nachträglich mit der Methode **setMessage** gesetzt und mit der Methode **getMessage** ausgelesen werden.
- Die Methode **PrintStackTrace** gibt folgende Informationen auf der Console aus:
  - Exception-Typ
  - Die Message (Falls vorhanden)
  - Die Quellcode-Zeile in der der Fehler aufgetreten ist.
  - Der CallStack
- Siehe Manual für weitere Methoden:

http://docs.oracle.com/javase/7/docs/api/java/lang/Throwable.html

Throw und Throws-Anweisung:

- Die Throw-Anweisung dient dazu eine Exception zu „werfen“/auszulösen.
- Falls in einer Methode eine Exception ausgelöst wird hat man die Möglichkeit diese mit abzufangen.

```
Void method ()throws Exception {
    ...
    throw new Exception(„Exception aufgetreten“);
    ...
}
```

Wichtige Exceptions

- **NullPointerException:** Tritt auf wenn man einen Objektzeiger verwendet, ohne ihn zuvor mit einen Wert zu initialisieren. Sprich: Dieser noch null ist.
- **ClassNotFoundException:** Tritt auf wenn verwendete Klasse nicht ins Programm gelinkt wurde.
- **ClassCastException:** Tritt auf wenn man ein Objekt zu einem Typ casten will, welcher nicht dem eigenen Objekt-Typ entspricht.

Try- catch- Anweisung:

```
try{
    ... //Code der normal Ausgeführt wird
}
catch(Exception e){
    ... //Code der Ausgeführt wird wenn eine Fehler/Ausnahme auftritt
}
finally{
... wird ausgeführt falls die in den Klammern angegebene Exception nicht zutreffend ist für den entstandenen Fehler.
}
```

Übung:

- Erstellen Sie ein Programm namens ExceptionExercise.
- Erstellen Sie Ihre eigene Exception (diese soll von Exception erben)
- Erstellen Sie eine Klasse namens ExceptionExercise.
- Erstellen Sie in der neuen Klasse eine Methode namens showText. Diese hat keinen Rückgabewert und erhält ein „Object“ als Parameter. Berprüfen sie in der Methode ob es sich bei dem Objekt um einen String handelt wenn ja, geben Sie diesen auf der Konsole aus. Wenn nein werfen Sie eine Exception.
- Erstellen Sie eine Mainklasse. Rufen Sie in dieser 2mal showText auf. Einmal mit String und einmal mit einem anderen Typ.

Dokumentation:

- Für Java gibt es das Tool JavaDoc. Hiermit lässt sich für dass erstellte Programm eine Dokumentation generieren.
- Eine Dokumentation lässt sich in Eclipse wie folgt erstellen:

  Menu -> Project -> Generate Javadoc...
- Voraussetzung für eine erfolgreiches Generieren ist, dass in den Kommentaren eine spezielle Syntax verwendet wird. Mit @ leitet man einen JavaDoc Befehl ein. Der Umfang der Befehle, hängt davon ab was dokumentiert wir: Klassen, Methoden, Variablen.
- Ein JavaDoc-Kommentar lässt sich mit „ /** “ beginnen

Java Doc-Befehle

Tabelle 19.3: Die wichtigsten Dokumentationskommentare im Überblick

Kommentar	Beschreibung	Beispiel
`@param`	Beschreibung der Parameter	`@param a A Value.`
`@see`	Verweis auf ein anderes Paket, einen anderen Typ, eine andere Methode oder Eigenschaft	`@see java.util.Date @see java.lang.String#length()`
`@version`	Version	`@version 1.12`
`@author`	Schöpfer	`@author Christian Ullenboom`
`@return`	Rückgabewert einer Methode	`@return Number of elements.`
`@exception/@throws`	Ausnahmen, die ausgelöst werden können	`@exception NumberFormatException`
`{@link Verweis}`	Ein eingebauter Verweis im Text im Code-Font. Parameter wie bei @see	`{@link java.io.File}`
`{@linkplain Verweis}`	Wie {@link}, nur im normalen Font	`{@linkplain java.io.File}`
`{@code Code}`	Quellcode im Code-Zeichensatz – auch mit HTML-Sonderzeichen	`{@code 1 ist < 2}`
`{@literal Literale}`	Maskiert HTML-Sonderzeichen. Kein Code-Zeichensatz	`{@literal 1 < 2 && 2 > 1}`
`@category`	Für Java 7 oder 8 geplant: Vergabe einer Kategorie	`@category Setter`

Diese Tabelle sowie weitere Informationen erhalten Sie unter folgenden Link:

Hausaufgabe

- JavaDoc Kommentare ins eigene Programm schreiben.
- JavaDokumentation erstellen.
- Fragen für letzte Stunde sammeln.