Web Technologien

# Exceptions

**Übungsfragen:**

*Was sind Exceptions im Kontext von Java?*

*Vordefinierte Fehlerklassen?*

*Unterschied Exception und Error?*

*Wichtigste Methoden in Klasse Throwable?*

*Aufbau Exceptions?*

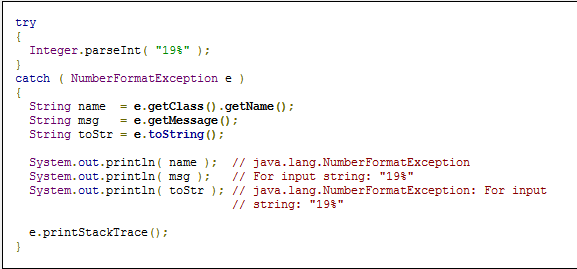
*Konstrukte: try, catch, finally, throw, throws?*

**Warum?** = Abfangen von Fehlern während Laufzeit von Programm

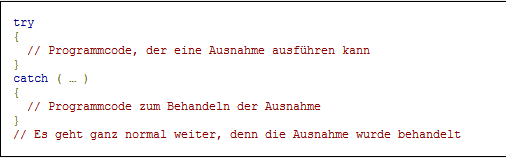
**Was?**

Ausnahmetypen

1. Checked Exception
   * Fehler zur Programmlaufzeit, die behandelt werden müssen, da Programmablauf sonst unterbrochen
2. Unchecked Exception (RunTimeExceptions)
   * Zur Laufzeit nicht behandelt 🡪 Löst keinen Programmcrash aus, nur Teil des Programmablaufs ist fehlerhaft
   * Können optional abgefangen werden
   * Beispiel: IllegalArgumentException, ArrayIndexOutOfBoundException, NullPointerException
3. Error vs. Exception
   * Abnormales Verhalten, auf JVM zurückzuführen
   * Kann zur Dokumentation oder Ausgabe Fehlermeldung abgefangen werden, sonst nicht
   * Beispiel: OutOfMemoryError -> Programm könnte Problem nicht beheben
4. Try Catch Aufbau
   * Try: Ausgeführter Code, der geprüft werden soll
   * Catch: Wirft Exception, Erzeugung eines Objektes, das direkt verarbeitet wird (z.B. Print einer Fehlermeldung) oder an Exception Klasse weitergeworfen wird
     + Informationen über übergebenes Exception Objekt lässt sich über Punktoperator abrufen
     + Werfen verschiedener Exception Objekte über mehrere Catch Blöcke möglich

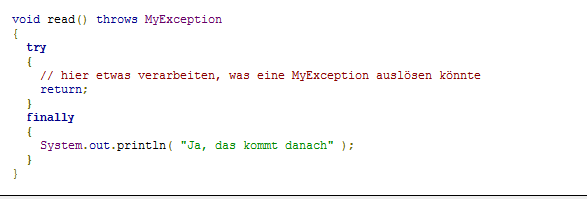


* + Wichtig: Wird ein Fehlerobjekt geworfen, wird der Programmablauf in try unterbrochen, also Code in try Block ab Fehlerauftritt nicht mehr ausgeführt!
    - Möglichkeit, Fehlerbehandlung in Catch Block zu machen und danach Programmablauf weiter auszuführen! (Vgl. Python proceedings mdate & key Tags)
    - Catch Klauseln nach Spezifität aufsteigend anordnen, da sonst bspw. Exception Anweisung jeden Fehler einfangen und weitere Klauseln ignoriert werden würden

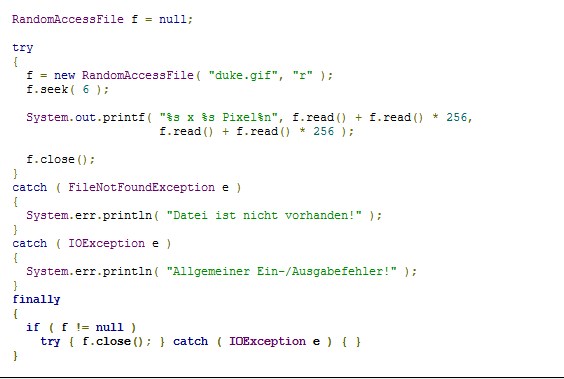


* Stack-Trace
  + Sammeln der Methoden in Stapel, die Fehlerobjekt ausgelöst haben (beginnend mit Methode in try Anweisung), um Fehler nachvollziehen zu können
  + Über printStackTrace() abrufbar
  + Ohne explizites Fehlerabfangen würde zur Laufzeit ebenfalls StackTrace per Default ausgegeben werden, daher try/catch Anweisung für explizite Fehlerbehandlung, sodass Programmlauf nicht unterbrochen!

1. Abschlussbehandlung in „finally“
   * Wird immer ausgeführt, auch wenn keine Exception geworfen wurde!
   * Anwendungsbeispiel: Schließen einer Datei sollte auf jeden Fall ausgeführt werden, unabhängig von Fehlerwurf!
   * Auch bei break, return oder continue in try Block
   * Ausnahme: system.exit() 🡪 Beenden des Programms!
   * Try Block zwingend erforderlich, catch Block nicht



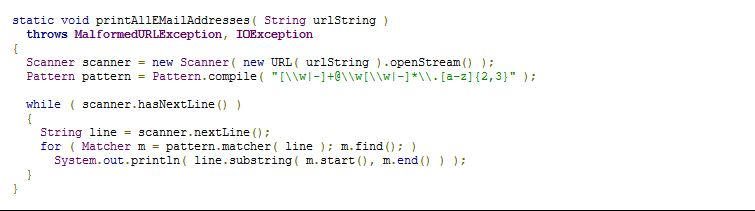
* + - Try immer notwendig, catch optional (dann throws in Methodenkopf deklarieren)



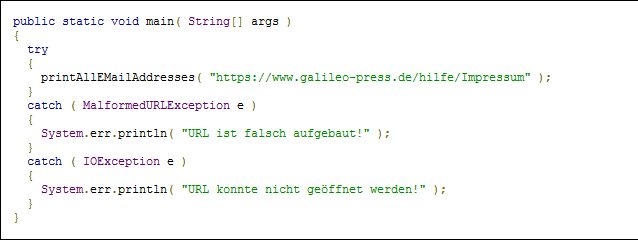
* + - Sollte ein Fehler beim Öffnen oder Auslesen der Datei passieren, würde close() übersprungen werden, daher Ausführung close() in finally Block!

1. Checked Exceptions mit throws im Methodenkopf
   * „Werfen“ eines Fehlers hin zum Aufruf der Methode und dortige Verarbeitung über try/catch
   * Nicht kontrollierte Behandlung von unchecked Exceptions, führen nicht zur Beendigung des Programms!
   * Behandlung von checked Exceptions
   * Programmablauf in aufgerufener Methode wird über throws zu Stelle in main() geworfen, die Methode aufruft und in main() behandelt
   * Verwendung wenn in main() Methoden aufgerufen und Fehler dieser abgefangen werden sollen
   * Weiterleitung und Behandlung in catch Block an anderer Stelle

Weiterleiten über throws in Deklaration der Methode:

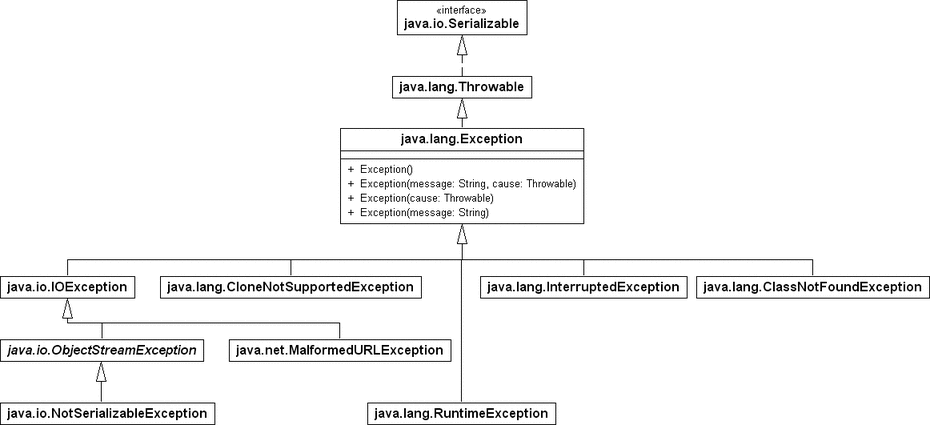


Werfen des Exception Objekts in main() wo Methode aufgerufen wird



* Fehlerweiterleitung an weitere Methoden möglich, sodass Kette an Methodenaufrufen Fehler weitergibt, bis catch Block Fehler behandelt

1. Unchecked Exceptions mit throw behandeln
   * Erzeugung eines Exception Objektes
   * Explizites Auslösen einer Fehlermeldung für Stellen, bei denen Programmablauf nicht gestört werden würde
   * Wird Exception Objekt erzeugt, wird Programmablauf in Methode beendet
   * Benötigt ebenfalls catch Block, der Fehler behandelt
2. Klassenhierarchie bei Exceptions
   * Alle Exceptionklassen erben von Exception, diese von Throwable und diese wiederum von Serializable



* + Problem: Wenn nur oberste Hierarchie an Exceptions gefangen werden würde, könnten verschiedene Ausnahmen in Programmablauf nicht unterschieden, analysiert und separat behandelt werden
  + Beachte: Hierarchisch gleichgestellte Exception Klassen explizit angeben, da sonst nicht über Vererbungsstruktur mitgenommen!

Generics

**Übungsfragen:**

*Welche Vorteile bringen Generics in Java Programmen?*

*Syntax von Generics?*

*Was sind Collections? Was sind bekannte Collections Klassen in Java?*

**Warum?**

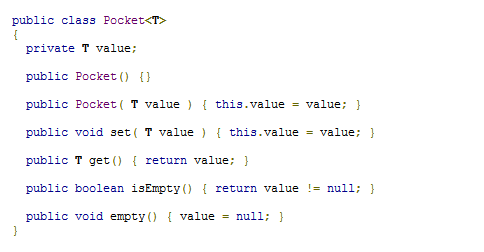
„Platzhalter“ für Deklaration des Datentyps eines Objekts über Typenparameter (Diamondoperator) bei Erzeugung des Objekts, sodass Datentyp flexibel bei neuem Objekt gesetzt werden kann 🡪 kein Cast benötigt bzw. nach Erzeugung des Objekts nachträglich möglich!

Verhindern von ClassCastExceptions zur Laufzeit, wenn Cast nicht möglich

Problem: Nicht ausführbare Casts von Compiler nicht erkannt!

**Was?**

1. Setzen der Platzhalter über Großbuchstaben bei Klassendeklaration die Generics verwenden soll



* + - Ersatz von „Object“ durch „T“ (Typenvariable für generische Typendeklaration)
    - Verwendung anderer Bezeichnungen ungünstig, da Referenz zu Klasse bezogen werden könnte
    - Deklaration mehrerer Typenparameter möglich <T, V>

1. Erzeugung der Objekte und setzen des Datentyps über Typenparameter



* + - Funktionen in Klassen der neuen Objekte die mit generischer Typdeklaration versehen wurden, werden nun alle in Integer bzw. String gecastet

Merke:

* Typenvariablen sind keine Datentypen, sondern Platzhalter

Setzen mehrerer Typenparameter:

* Methoden in Klassendeklarationen verschiedenen Datentypen bei Objekterzeugung zuweisen

Generische Methoden:

* Verwendung von Typenparametern in Methoden, um unabhängig von späterer Datentypzuweisung bei Objekterzeugung Vorgänge bearbeiten zu können

Begrenzte Typenparameter:

* Bei Deklaration der Typenparameter kann über „extends“ bestimmter Datentyp spezifiziert werden
  + - Spätere Objekterzeugung erlaubt nur Zuweisung von Datentyp der in Zuweisung erlaubt
    - Beachte: Bei Verwendung in Typendeklaration von Klassen müssen Methoden in Klasse Eingrenzung ebenfalls verwenden

Vererbung bei Generics:

* In <> gesetzte Typenparameter können bei Datentypen Zuweisung in Objekterzeugung keine Vererbungsstrukturen übernehmen
* Beispiel Test<Number> kann bei Objekterzeugung nicht in Test<Integer> zugewiesen werden
* Test<Integer> ist kein Subtyp von Test<Number>!
* Methodenaufruf für erzeugte Instanz kann Vererbungsstruktur (Bspw. Integer von Number) jedoch verwenden!

1. Geschachtelte Klassen
2. HTML & CSS
3. Serverseitige Konzepte (Mapper, Verbindung DB und Web Server)
4. Git