

9. Tutorium Java-API, Prinzipien der ObjektOrientierung

Tutorium 14

Péter Bohner | 18.01.2023



Tutorium 14

023-01-18





Tutorium 14 ─Wiederholung

─Wiederholung

Welche der Aussagen treffen auf Interfaces zu?

Welche der Aussagen treffen auf Interfaces zu?













Übungsblatt 4

JavaDoc

OO-Design-Prinzipien

Wiederholung

•00 18.01.2023

000000 Péter Bohner: Tutorium 14

00000

Exceptions 000000000000000

Java-API 000000000



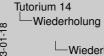
Welche der Aussagen treffen auf Interfaces zu?

- Definieren eine Schnittstelle
- Definieren eine Sammlung von Methodensignaturen

Péter Bohner: Tutorium 14

- Eine Klasse kann mehrere Interfaces implementieren
- Ein Interface kann von beliebig vielen anderen Interfaces erben

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions •00 000000 00000 000000000 0000000000





 Eine Klasse kann mehrere Interfaces implementieren Ein Interface kann von beliebig vielen anderen Interfaces erben └─Wiederholung



Welche der Aussagen treffen auf Interfaces zu?

- Definieren eine Schnittstelle
- Definieren eine Sammlung von Methodensignaturen
- Eine Klasse kann mehrere Interfaces implementieren
- Ein Interface kann von beliebig vielen anderen Interfaces erben

Was stellt eine Exception dar?

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions •00 000000 00000 000000000 0000000000

Tutorium 14 -Wiederholung

Welche der Aussagen treffen auf Interfaces zu? Definieren eine Schnittstelle

 Definieren eine Sammlung von Methodensignaturen Was stellt eine Exception dar?

 Eine Klasse kann mehrere Interfaces implementieren Ein Interface kann von beliebig vielen anderen Interfaces erben

└─Wiederholung



Welche der Aussagen treffen auf Interfaces zu?

- Definieren eine Schnittstelle
- Definieren eine Sammlung von Methodensignaturen
- Eine Klasse kann mehrere Interfaces implementieren
- Ein Interface kann von beliebig vielen anderen Interfaces erben

Was stellt eine Exception dar?

Eine Ausnahme

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions •00 000000 00000 000000000 0000000000

Tutorium 14 -Wiederholung Wiederholung

Welche der Aussagen treffen auf Interfaces zu? Definieren eine Schnittstelle

Definieren eine Sammlung von Methodensignaturen

Eine Ausnahme

 Eine Klasse kann mehrere Interfaces implementieren Ein Interface kann von beliebig vielen anderen Interfaces erben Was stellt eine Exception dar?



Welche der Aussagen treffen auf Interfaces zu?

- Definieren eine Schnittstelle
- Definieren eine Sammlung von Methodensignaturen
- Eine Klasse kann mehrere Interfaces implementieren
- Ein Interface kann von beliebig vielen anderen Interfaces erben

Was stellt eine Exception dar?

Eine Ausnahme

Mit welchen Schlüsselwörtern wirft man in Java eine Exception?

Wiederholung Übungsblatt 4 Java-API OO-Design-Prinzipien JavaDoc Exceptions •00 000000 00000 000000000 0000000000 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 -Wiederholung -Wiederholung

Definieren eine Schnittstelle

Definieren eine Sammlung von Methodensignaturen

 Eine Klasse kann mehrere Interfaces implementieren Ein Interface kann von beliebig vielen anderen Interfaces erben Was stellt eine Exception dar?

Mit welchen Schlüsselwörtern wirft man in Java eine Exception?



Welche der Aussagen treffen auf Interfaces zu?

- Definieren eine Schnittstelle
- Definieren eine Sammlung von Methodensignaturen
- Eine Klasse kann mehrere Interfaces implementieren
- Ein Interface kann von beliebig vielen anderen Interfaces erben

Was stellt eine Exception dar?

Eine Ausnahme

Mit welchen Schlüsselwörtern wirft man in Java eine Exception?

throw new ExceptionName();

Wiederholung Übungsblatt 4 Java-API OO-Design-Prinzipien JavaDoc Exceptions •00 000000 00000 000000000 0000000000 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 -Wiederholung -Wiederholung

Welche der Aussagen treffen auf Interfaces zu? Definieren eine Schnittstelle

Definieren eine Sammlung von Methodensignaturen

 Eine Klasse kann mehrere Interfaces implementieren Ein Interface kann von beliebig vielen anderen Interfaces erben

Was stellt eine Exception dar? Mit welchen Schlüsselwörtern wirft man in Java eine Exception?

throw new ExceptionName():



Wie fange und behandle ich eine Exception?

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API 000 000000 00000 000000000000000 000000000 0000000000

Programmieren Tutorium

─Wiederholung

Tutorium 14

Wie fange und behandle ich eine Exception?



2023-01-18







Péter Bohner: Tutorium 14





18.01.2023

Wiederholung

18.01.2023

000



```
Wie fange und behandle ich eine Exception?
```

Übungsblatt 4

Péter Bohner: Tutorium 14

000000

JavaDoc

00000

Exceptions

000000000000000

```
try {
    // code
 catch (ExceptionType e) {
    // exception-handling
```

OO-Design-Prinzipien 0000000000

Java-API

000000000

Programmieren Tutorium



Tutorium 14 ─Wiederholung

2023-01-18





Wiederholung

18.01.2023



```
Wie fange und behandle ich eine Exception?
try {
```

Übungsblatt 4

Péter Bohner: Tutorium 14

000000

```
// code
catch (ExceptionType e) {
  // exception-handling
```

Welche der Aussagen treffen idealerweise auf Exceptions zu?

JavaDoc

00000

Exceptions

000000000000000

OO-Design-Prinzipien 0000000000

Java-API

000000000

Programmieren Tutorium



─Wiederholung ─Wiederholung

Tutorium 14



Welche der Aussagen treffen idealerweise auf Exceptions zu:



Wie fange und behandle ich eine Exception?

```
try {
    // code
} catch (ExceptionType e) {
    // exception-handling
}
```

Welche der Aussagen treffen idealerweise auf Exceptions zu?

- Steuern niemals den Kontrollfluss
- Stellen die Ausnahme dar
- Fangen vorhersehbare und behandelbare Fehler ab

Trennen Programmlogik und Fehlerbehandlung sauber voneinander

 Tutorium 14

Wiederholung

Wiederholung

Wiederholung

Wie fange und behandle ich eine Exception? try { // code

} catch (ExceptionType e) {

Welche der Aussagen treffen idealerweise auf Exceptions zuf

Steuern niemets den Kontrolliuss

Steuern niemals den Kontrollfus
 Stellen die Ausnahme der

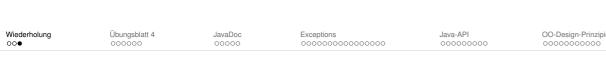
Stellen die Ausnahme der
 Fangen vorherseltbare und behandelbare Fehler ab

Trennen Programmlogik und Fehlerbehandlung sauber voneinander



Welche der Aussagen treffen auf Rekursive-Methoden zu?

Péter Bohner: Tutorium 14



─Wiederholung

Tutorium 14 ─Wiederholung

Welche der Aussagen treffen auf Rekursive-Methoden zu?

OO-Design-Prinzipien

Programmieren Tutorium

18.01.2023



Welche der Aussagen treffen auf Rekursive-Methoden zu?

- Speicher-/Zeitaufwändiger als iteratives Äquivalent
- Termination kann nicht vorausgesagt werden
- Vorteilhaft, falls Anzahl der Iterationen noch nicht bekannt
- Falls bekannt, besser iterativ



Tutorium 14 ─Wiederholung

Welche der Aussagen treffen auf Rekursive-Methoden zu? Speicher-Zeitaufwändiger als iteratives Äquivalent Termination kann nicht vorausgesagt werden

 Vorteilhalt, falls Anzahl der Iterationen noch nicht bekannt Falls bekannt, besser iterativ

─Wiederholung



Welche der Aussagen treffen auf Rekursive-Methoden zu?

- Speicher-/Zeitaufwändiger als iteratives Äquivalent
- Termination kann nicht vorausgesagt werden
- Vorteilhaft, falls Anzahl der Iterationen noch nicht bekannt
- Falls bekannt, besser iterativ

Was passiert in Java, wenn eine Methode aufgerufen wird?

Übungsblatt 4 Wiederholung Java-API OO-Design-Prinzipien JavaDoc Exceptions 000000 00000 000000000 0000000000 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14 -Wiederholung

 Termination kann nicht vorausgesagt werden Vorteilhalt, falls Anzahl der Iterationen noch nicht bekannt

Falls bekannt, besser iterativ

Was passiert in Java, wenn eine Methode aufgerufen wird?

Welche der Aussagen treffen auf Rekursive-Methoden zu? Speicher-/Zeitaufwändiger als iteratives Äguivalent

-Wiederholung



Welche der Aussagen treffen auf Rekursive-Methoden zu?

- Speicher-/Zeitaufwändiger als iteratives Äquivalent
- Termination kann nicht vorausgesagt werden
- Vorteilhaft, falls Anzahl der Iterationen noch nicht bekannt
- Falls bekannt, besser iterativ

Was passiert in Java, wenn eine Methode aufgerufen wird?

- Lokale Variablen (inkl. Paramter),
- Operanden,
- Rücksprungadresse
 - ... werden auf den Aufrufstapel geschrieben

Wie deklariert man in Java die Verwendung eines Generics T in einer Klasse?



Tutorium 14 -Wiederholung -Wiederholung

Welche der Aussagen treffen auf Rekursive-Methoden zu? Speicher-/Zeitaufwändiger als iteratives Äquivalent

Termination kann night unravaneaart wenden

Vorteilhaft, falls Anzahl der Iterationen noch nicht bekannt

Falls bekannt, besser iterativ

Was passiert in Java, wenn eine Methode aufgerufen wird? Lokale Variablen (inkl. Paramter).

 Operanden. Rücksprungadresse

... werden auf den Aufrufstapel geschrieben Wie deklariert man in Java die Verwendung eines Generics T in einer Klasse:



Welche der Aussagen treffen auf Rekursive-Methoden zu?

- Speicher-/Zeitaufwändiger als iteratives Äquivalent
- Termination kann nicht vorausgesagt werden
- Vorteilhaft, falls Anzahl der Iterationen noch nicht bekannt
- Falls bekannt, besser iterativ

Was passiert in Java, wenn eine Methode aufgerufen wird?

Lokale Variablen (inkl. Paramter),

000000

- Operanden,
- Rücksprungadresse
 - ... werden auf den Aufrufstapel geschrieben

Wie deklariert man in Java die Verwendung eines Generics T in einer Klasse? public class ClassName<T> { ... }

00000

Wie gibt man den konkreten Datentyp bei der Instanzierung einer Klasse die Generics nutzt an?

9		- 4.0			• ·······
Wiederholung	Übungsblatt 4	JavaDoc	Exceptions	Java-API	OO-Design-P



0000000000000000



Wie deklariert man in Java die Verwendung eines Generics T in einer Klasser geblig class Classiane (T) (, , ,)

Wie gibt man den konkreten Datentyp bei der Instanzierung einer Klasse die Generica nutzt an



Welche der Aussagen treffen auf Rekursive-Methoden zu?

- Speicher-/Zeitaufwändiger als iteratives Äquivalent
- Termination kann nicht vorausgesagt werden
- Vorteilhaft, falls Anzahl der Iterationen noch nicht bekannt
- Falls bekannt, besser iterativ

Was passiert in Java, wenn eine Methode aufgerufen wird?

- Lokale Variablen (inkl. Paramter),
- Operanden,
- Rücksprungadresse
 - ... werden auf den Aufrufstapel geschrieben

Wie deklariert man in Java die Verwendung eines Generics T in einer Klasse?

public class ClassName<T> { ... }

Wie gibt man den konkreten Datentyp bei der Instanzierung einer Klasse die Generics nutzt an?

ClassName<String> objectName = new ClassName<>():

derholung	Übungsblatt 4	JavaDoc	Exceptions	Java-API	OO-Design-Prinzipien
•	000000	00000	00000000000000	00000000	0000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 ─Wiederholung

-Wiederholuna

Welche der Aussagen treffen auf Rekursive-Methoden zu? Speicher-/Zeitaufwändiger als iteratives Äguivalent

Termination kann night unravanes art wenden

Falls bekannt, besser iterativ

Was passiert in Java, wenn eine Methode aufgerufen wird? Lokale Variablen (inkl. Panamter),

 Operanden. Rücksprungadresse

... werden auf den Aufrufstapel geschrieben

Wie deklariert man in Java die Verwendung eines Generics T in einer Klasse:

public class ClassEgme(T) (...) Wie gibt man den konkreten Datentyp bei der Instanzierung einer Klasse die Generics nutzt an ClassName-String- objectName = new ClassName->():



Allgemein

A 9 und B 15 Abgaben

Exceptions 000 •00000 00000 000000000000000 000000000 0000000000

Tutorium 14 Übungsblatt 4 Übungsblatt 4

Aligemein

A 9 und B 15 Abgaben

Übungsblatt 4 JavaDoc Java-API OO-Design-Prinzipien Wiederholung

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

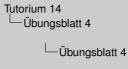


Allgemein

18.01.2023

- A 9 und B 15 Abgaben
- Die meisten sollten ihren Übungsschein haben

Péter Bohner: Tutorium 14







Allgemein

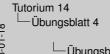
18.01.2023

- A 9 und B 15 Abgaben
- Die meisten sollten ihren Übungsschein haben

Péter Bohner: Tutorium 14

Logik Hardcoden bringt nichts!

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000 •00000 00000 000000000000000 000000000 0000000000









Allgemein

000

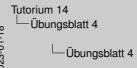
18.01.2023

- A 9 und B 15 Abgaben
- Die meisten sollten ihren Übungsschein haben

Péter Bohner: Tutorium 14

■ Logik Hardcoden bringt nichts! Wieso muss ich das beim 4. ÜB noch sagen?

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien •00000 00000 000000000000000 000000000 0000000000







Allgemein

- A 9 und B 15 Abgaben
- Die meisten sollten ihren Übungsschein haben

Péter Bohner: Tutorium 14

■ Logik Hardcoden bringt nichts! Wieso muss ich das beim 4. ÜB noch sagen? Wenn ihr z.B. den BFS nicht hin bekommt, wird das mit den Abschlussaufgaben schwer.

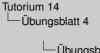
Code Stil

18.01.2023

Leere Zeilen zwischen Abschnitten (oder besser in Hilfsmethoden auslagern)

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions 000 •00000 00000 000000000 0000000000

Programmieren Tutorium





Leere Zeilen zwischen Abschnitten (oder besser in Hilfsmethoden auslagern)

Übungsblatt 4



Allgemein

- A 9 und B 15 Abgaben
- Die meisten sollten ihren Übungsschein haben
- Logik Hardcoden bringt nichts! Wieso muss ich das beim 4. ÜB noch sagen?

 Wenn ihr z.B. den BFS nicht hin bekommt, wird das mit den Abschlussaufgaben schwer.

Code Stil

- Leere Zeilen zwischen Abschnitten (oder besser in Hilfsmethoden auslagern)
- Bedingung und Block nicht auf eine Zeile.

2 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14
Übungsblatt 4
Übungsblatt 4

A 2 and B 15 Abgaban
Dis meisten sollten ihren Übungsschein haben
Dis meisten sollten ihren Übungsschein haben
Logik Hardcoden bringt nichtst ^{West zwiss üh das bein 4.00 zuch agen?}
Wenn ihr z.B. den BFS nicht hin bekommt, wird das mit den Abschlussas

Leere Zeiten zwischen Abschnitten (oder besser in Hilfsmethoden auslagern)
 Bedingung und Block richt auf eine Zeite.



Allgemein

- A 9 und B 15 Abgaben
- Die meisten sollten ihren Übungsschein haben
- Logik Hardcoden bringt nichts! Wieso muss ich das beim 4. ÜB noch sagen? Wenn ihr z.B. den BFS nicht hin bekommt, wird das mit den Abschlussaufgaben schwer.

Code Stil

- Leere Zeilen zwischen Abschnitten (oder besser in Hilfsmethoden auslagern)
- Bedingung und Block nicht auf eine Zeile.
- Haut einfach den Auto-Formatter vor der Abgabe drauf



Tutorium 14 Übungsblatt 4

Übungsblatt 4

 A 9 und B 15 Abgaben Logik Hardcoden bringt nichts! West muss ich das beim 4. Dit noch sagen?

Die meisten sollten ihren Übungsschein haben

Bedingung und Block nicht auf eine Zeile.

Haut einfach den Auto-Formatter vor der Abosbe drauf



Allgemein

- A 9 und B 15 Abgaben
- Die meisten sollten ihren Übungsschein haben
- Logik Hardcoden bringt nichts! Wieso muss ich das beim 4. ÜB noch sagen? Wenn ihr z.B. den BFS nicht hin bekommt, wird das mit den Abschlussaufgaben schwer.

Code Stil

- Leere Zeilen zwischen Abschnitten (oder besser in Hilfsmethoden auslagern)
- Bedingung und Block nicht auf eine Zeile.
- Haut einfach den Auto-Formatter vor der Abgabe drauf
- Achtet auf die Rechtschreibung, vermeidet Stilblüten

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API •00000

Tutorium 14 Übungsblatt 4

Übungsblatt 4

 A 9 und B 15 Abgaben Die meisten sollten ihren Übungsschein haben Logik Hardcoden bringt nichts! West muss ich das beim 4. Dit noch sagen?

Bedingung und Block nicht auf eine Zeile.

Haut einfach den Auto-Formatter vor der Abosbe drauf

Achtet auf die Rechtschreibung, vermeidet Stilblüten.



Javadoc

Beschreibt die Funktionalität von Klassen und (öffentlichen) Methoden / Attribute

Wiederholung 000

Übungsblatt 4 000000

JavaDoc 00000

Exceptions 000000000000000 Java-API 000000000 OO-Design-Prinzipien 0000000000

Tutorium 14 Übungsblatt 4 Übungsblatt 4

Beschreibt die Funktionalität von Klassen und (öffentlichen) Methoden / Attribute



Javadoc

- Beschreibt die Funktionalität von Klassen und (öffentlichen) Methoden / Attribute
- Nur mit Hilfe der Javadoc soll klar sein, was genau das dokumentierte Objekt soll

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000 000000 00000 000000000000000 000000000 0000000000

Tutorium 14 Übungsblatt 4 Übungsblatt 4

Nur mit Hilfe der Javadoc soll klar sein, was genau das dokumentierte Objekt soll



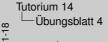
Javadoc

18.01.2023

- Beschreibt die Funktionalität von Klassen und (öffentlichen) Methoden / Attribute
- Nur mit Hilfe der Javadoc soll klar sein, was genau das dokumentierte Objekt soll
- Deshalb: (in-) Valide Parameter (Kombinationen) beschreiben

Péter Bohner: Tutorium 14

Programmieren Tutorium





Deshalb: (in-) Valide Parameter (Kombinationen) beschreiben

Übungsblatt 4



Javadoc

- Beschreibt die Funktionalität von Klassen und (öffentlichen) Methoden / Attribute
- Nur mit Hilfe der Javadoc soll klar sein, was genau das dokumentierte Objekt soll
- Deshalb: (in-) Valide Parameter (Kombinationen) beschreiben
- Beschreiben, was in Randfällen, Fehlerzuständen passiert

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions 000 000000 00000 000000000 0000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 Übungsblatt 4

Beschreiben, was in Randfällen, Fehlerzuständen passiert

Deshab: (in-) Valide Parameter (Kombinationen) beschreiben

Übungsblatt 4



Javadoc

- Beschreibt die Funktionalität von Klassen und (öffentlichen) Methoden / Attribute
- Nur mit Hilfe der Javadoc soll klar sein, was genau das dokumentierte Objekt soll
- Deshalb: (in-) Valide Parameter (Kombinationen) beschreiben
- Beschreiben, was in Randfällen, Fehlerzuständen passiert
- Welche Exceptions werden wann geworfen?

Übungsblatt 4 Java-API OO-Design-Prinzipien Wiederholung JavaDoc Exceptions 000 000000 00000 000000000 0000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 Übungsblatt 4

> Nur mit Hilfe der Javadoc soll klar sein, was genau das dokumentierte Objekt soll Übungsblatt 4 Deshab: (in-) Valide Parameter (Kombinationen) beschreiben

Beschreiben, was in Randfällen, Fehlerzuständen passiert

Welche Exceptions werden wann geworfen?

18.01.2023

Péter Bohner: Tutorium 14



Trennung von UI und Model Was gehört in die UI?

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API 000 000000 00000 000000000000000 000000000 0000000000

Programmieren Tutorium

Tutorium 14 Übungsblatt 4 Übungsblatt 4 - Aufgabe A

Trennung von UI und Model

Was gehört in die UI?



Trennung von UI und Model

Was gehört in die UI?

18.01.2023

Ein und Ausgabe (System.out, Scanner)

Péter Bohner: Tutorium 14

- Das Parsen von Eingaben
- Die Serialisation von Daten, die Ausgegeben werden sollen



Programmieren Tutorium





Ein und Ausgabe (System.out, Scanner)

Das Parsen von Eingaben

Die Serialisation von Daten, die Ausgebeben werden sollen



Trennung von UI und Model

- Was gehört in die UI?
- Ein und Ausgabe (System.out, Scanner)
- Das Parsen von Eingaben
- Die Serialisation von Daten, die Ausgegeben werden sollen
- Was gehört ins Model?



2 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14

—Übungsblatt 4

—Übungsblatt 4 - Aufgabe A



- Ein und Ausgabe (System.out, Scanner)
 Das Parsen von Eingaben
- Die Serialisation von Daten, die Ausgegeben werden sollen
 - Die Serialisation von Daten, die Ausgegeben wer
 Was gehört ins Model?



Trennung von UI und Model

- Was gehört in die UI?
- Ein und Ausgabe (System.out, Scanner)
- Das Parsen von Eingaben
- Die Serialisation von Daten, die Ausgegeben werden sollen
- Was gehört ins Model?
- Die Modellierung des wahren Zustandes und Logik
- Keine Ein und Ausgaben!

Übungsblatt 4 Wiederholung Java-API OO-Design-Prinzipien JavaDoc Exceptions 000 000000 00000 000000000 0000000000

Tutorium 14 Übungsblatt 4

Übungsblatt 4 - Aufgabe A

 Was gehört in die UI? Ein und Ausgabe (System.out, Scanner)

 Das Parsen von Eingaben Die Serialisation von Daten, die Ausgegeben werden sollen

 Was gehört ins Model? Die Modellierung des wahren Zustandes und Logik

Keine Ein und Ausgeben!



Trennung von UI und Model

- Was gehört in die UI?
- Ein und Ausgabe (System.out, Scanner)
- Das Parsen von Eingaben
- Die Serialisation von Daten, die Ausgegeben werden sollen
- Was gehört ins Model?
- Die Modellierung des wahren Zustandes und Logik
- Keine Ein und Ausgaben!
- Das Model sollte keine invaliden Daten darstellen (nur richtig übertragenes Terrain sollte auch in einem Terrain-Objekt gespeichert werden)



Tutorium 14 Übungsblatt 4

Übungsblatt 4 - Aufgabe A

Ein und Ausgabe (System.out, Scanner)

 Das Parsen von Eingaben Die Serialisation von Daten, die Ausgebehen werden sollen

Was gehört ins Model?

 Die Modellierung des wahren Zustandes und Logik Keine Ein und Ausgeben!

Das Model sollte keine invaliden Daten darstellen (nur richtig übertragenes Terrain sollte auch in einem

Terrain-Objekt gespeichert werden)

18.01.2023



Häufige Fehler in Aufgabe A

Code-Duplication statt Hilfsmethoden parametrisieren

Péter Bohner: Tutorium 14

Programmieren Tutorium

Tutorium 14

Übungsblatt 4

Übungsblatt 4

HSudge Fehler in Aufgabe A

© Cath Diglication start tritimentodes parametriaren



Häufige Fehler in Aufgabe A

- Code-Duplication statt Hilfsmethoden parametrisieren
- Ul und IO vermischen

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000000 00000 000000000000000 000000000 0000000000

Programmieren Tutorium

Tutorium 14 Übungsblatt 4 Übungsblatt 4

Häufige Fehler in Aufgabe A

Code-Duplication statt Hilfsmethoden parametrisieren



Häufige Fehler in Aufgabe A

- Code-Duplication statt Hilfsmethoden parametrisieren
- Ul und IO vermischen
- Schlechte OO-Modellierung

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000 000000 00000 000000000000000 000000000 0000000000

Tutorium 14 Übungsblatt 4

Häufige Fehler in Aufgabe A

Code-Duplication statt Hilfsmethoden parametrisieren Ul und IO vermischen Schlechte OO-Modellierung

Übungsblatt 4



Häufige Fehler in Aufgabe A

- Code-Duplication statt Hilfsmethoden parametrisieren
- Ul und IO vermischen
- Schlechte OO-Modellierung
- Kontrollfluss mit Exceptions

Häufige Fehler in Aufgabe B

Unvollständige JavaDoc (z.B. @throws fehlt)

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions 000 000000 00000 000000000000000 000000000 0000000000

Programmieren Tutorium

Tutorium 14 Übungsblatt 4 Übungsblatt 4

 Schlechte OO-Modellierung Kontrollfluss mit Exceptions Unvolständige JavaDoc (z.B. @throws fehlt)

18.01.2023

Péter Bohner: Tutorium 14



Häufige Fehler in Aufgabe A

- Code-Duplication statt Hilfsmethoden parametrisieren
- Ul und IO vermischen
- Schlechte OO-Modellierung
- Kontrollfluss mit Exceptions

Häufige Fehler in Aufgabe B

- Unvollständige JavaDoc (z.B. @throws fehlt)
- Falscher Exception-Typ

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions 000 000000 00000 000000000 0000000000

Übungsblatt 4

Tutorium 14 Übungsblatt 4

 Code-Duplication statt Hillsmethoden parametrisieren UI und IO vermischen Schlechte OO-Modellierung Kontrollfluss mit Exceptions Häufige Fehler in Aufgabe B

Unvollständige JavaDoc (z.B. @throws fehlt)

Falscher Exception-Typ



NullPointerException

Thrown when an application attempts to use null in a case where an object is required.

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14

Übungsblatt 4

Übungsblatt 4 - Aufgabe B Exceptions







Thrown when an application attempts to use null in a case where an object is required.

IllegalStateException

Signals that a method has been invoked at an illegal or inappropriate time. In other words, the Java environment or Java application is not in an appropriate state for the requested operation.

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions 000000 00000 000000000 0000000000

Tutorium 14 Übungsblatt 4

Übungsblatt 4 - Aufgabe B Exceptions

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium



Tutorium 14 Übungsblatt 4

Übungsblatt 4 - Aufgabe B Exceptions

NullPointerException

Thrown when an application attempts to use null in a case where an object is required.

IllegalStateException

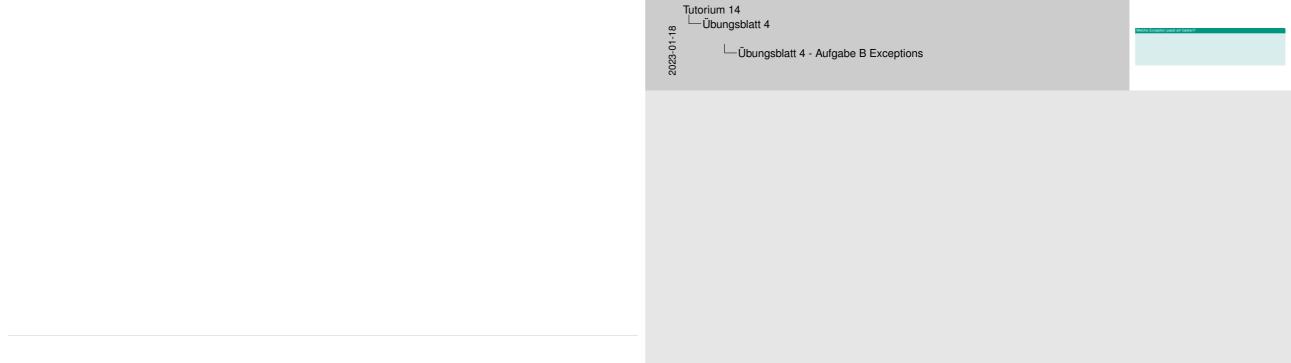
Signals that a method has been invoked at an illegal or inappropriate time. In other words, the Java environment or Java application is not in an appropriate state for the requested operation.

IllegalArgumentException

Thrown to indicate that a method has been passed an illegal or inappropriate argument.

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions 000000 00000 0000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium





```
Welche Exception passt am besten?
 if(sweets == null) throw new Something();
```

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 00000 00000 000000000000000 000000000 0000000000

Programmieren Tutorium

Tutorium 14 Übungsblatt 4

Welche Exception passt am besten?

• if(seets = null) throw new Something();

Übungsblatt 4 - Aufgabe B Exceptions

000

18.01.2023

Péter Bohner: Tutorium 14



```
Welche Exception passt am besten?
 if(sweets == null) throw new Something();
    if(sweets == null) throw new NullPointerException("sweets Argument may not be null.");
```



Tutorium 14 Übungsblatt 4 Übungsblatt 4 - Aufgabe B Exceptions



Wiederholung

18.01.2023

Péter Bohner: Tutorium 14

Übungsblatt 4

00000

Programmieren Tutorium



```
Welche Exception passt am besten?
```

```
if(sweets == null) throw new Something();
if(sweets == null) throw new NullPointerException("sweets Argument may not be null.");
```

if(sweets == null) throw new NullPointerException("sweets Argument may not be null.");

if(day < 0 || day > NUMBER_OF_DAYS) throw new Something();

Tutorium 14

Übungsblatt 4

Übungsblatt 4 - Aufgabe B Exceptions

Péter Bohner: Tutorium 14



```
Welche Exception passt am besten?
```

18.01.2023

```
if(sweets == null) throw new Something();
if(sweets == null) throw new NullPointerException("sweets Argument may not be null.");
if(day < 0 || | day > NUMBER OF DAYS) throw new Something();
```

if(day < 0 || day > NUMBER_OF_DAYS) throw new Something();
if(day < 0 || day > NUMBER_OF_DAYS) throw new IllegalArgumentException("Day is out of range.");

Programmieren Tutorium

Tutorium 14

Übungsblatt 4

Übungsblatt 4 - Aufgabe B Exceptions

if(weets = mall) three was Zombitze(); if(weets = mall) the same well-based extractoring framets Argument may not if(almy < mall) the XDMER_0F_DBY3 three was Zomething(); if(almy < mall | day > NDMER_0F_DBY3 three was Integribe passet Exception(10)

if(!canOpenDoor(day)) throw new Something();



```
Welche Exception passt am besten?
```

```
if(sweets == null) throw new Something();
  if(sweets == null) throw new NullPointerException("sweets Argument may not be null.");
■ if(day < 0 || day > NUMBER_OF_DAYS) throw new Something();
  if(day < 0 || day > NUMBER_OF_DAYS) throw new IllegalArgumentException("Day is out of range.");
```

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API 00000

Tutorium 14 Übungsblatt 4 Übungsblatt 4 - Aufgabe B Exceptions

if(day < 0 || day > MEMBER_OF_DAYS) throw new Something(); if(day < 0 || day > MEMBER OF DAYS) throw new IllegalArgumentException('Day is out of range



```
Welche Exception passt am besten?
```

```
if(sweets == null) throw new Something();
if(sweets == null) throw new NullPointerException("sweets Argument may not be null.");
if(day < 0 || day > NUMBER_OF_DAYS) throw new Something();
```

if(day < 0 || day > NUMBER_OF_DAYS) throw new IllegalArgumentException("Day is out of range.");

if(!canOpenDoor(day)) throw new Something();
if(!canOpenDoor(day)) throw new IllegalStateException("The door may not be opened yet");



Programmieren Tutorium

Tutorium 14

Übungsblatt 4

Übungsblatt 4 - Aufgabe B Exceptions

Woode Exception pasts are bestor?

If some time and 10 time and commission;
if some time and 10 time are distributed;
if streams = mail 10 time are distributed;
if streams = mail 10 time are distributed are distributed as a mail 10 time are distributed;
if streams = 10 time are distributed as a mail time are distribu

— Ubungsblatt 4 - Aufgabe B Exceptions



```
Welche Exception passt am besten?
```

```
if(sweets == null) throw new Something();
  if(sweets == null) throw new NullPointerException("sweets Argument may not be null.");
```

- if(day < 0 || day > NUMBER_OF_DAYS) throw new Something();
- if(day < 0 || day > NUMBER_OF_DAYS) throw new IllegalArgumentException("Day is out of range."); if(!canOpenDoor(day)) throw new Something();
 - if(!canOpenDoor(day)) throw new IllegalStateException("The door may not be opened yet");
- Anmerkung: Error messages bitte als Konstanten



Tutorium 14 Übungsblatt 4

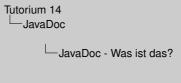
Übungsblatt 4 - Aufgabe B Exceptions

if(day < 0 || day > NUMBER_OF_DAYS) throw new Something(); if(day < 0 || day > MDMDER_OF_DAYS) throw new IllegalArgumentException('Day is out of ran if(|canOpenDoor(day)) throw new Something();

Anmerkung: Error messages bitte als Konstanten



2023-01-18





Javadoc...

... beschreibt Klassen, Methoden, Konstuktoren und Felder

JavaDoc

•0000

Exceptions

000000000000000

Java-API

000000000

- ... wird direkt in den Code geschreiben
- ... erleichtert das Verständnis

0000000000

OO-Design-Prinzipien

Ö

Tutorium 14

JavaDoc

JavaDoc

Javadocu.

Losschreibt Risssen, Methoden, Konstuktoren und Felder

Losschreibt in den Code geschreiben

en erleichtert das Verständnis

2023-01

__JavaDoc - Was ist das?

Übungsblatt 4



Javadoc...

- ... beschreibt Klassen, Methoden, Konstuktoren und Felder
- ... wird direkt in den Code geschreiben
- ... erleichtert das Verständnis

Regeln und Ziele

- einheitlich auf Deutsch oder auf Englisch
- vollständige und aussagekräftige Beschreibung

Programmieren Tutorium

Tutorium 14

JavaDoc

JavaDoc - Was ist das?





Tutorium 14 -JavaDoc └─JavaDoc - Was ist das?

... beschreibt Klassen, Methoden, Konstuktoren und Felder ... wird direkt in den Code geschreiben ... erleichtert das Verständnis Regeln und Ziele

einheitlich auf Deutsch oder auf Englisch vollständige und aussagekräftige Beschreibung

Javadoc...

- ... beschreibt Klassen, Methoden, Konstuktoren und Felder
- ... wird direkt in den Code geschreiben
- ... erleichtert das Verständnis

Regeln und Ziele

- einheitlich auf Deutsch oder auf Englisch
- vollständige und aussagekräftige Beschreibung

Javadoc zur Java API

000

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions 000000 •0000 000000000 0000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium



2023-01-18





Wiederholung 000 13/52

18.01.2023

Péter Bohner: Tutorium 14

JavaDoc

00000

Übungsblatt 4

Programmieren Tutorium

Exceptions

000000000000000

Java-API



beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000 000000 00000 000000000000000 000000000 0000000000 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 -JavaDoc ☐ JavaDoc für Klassen

beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist



- beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist
- mögliche Verwendungszwecke können erwähnt werden



Programmieren Tutorium

Tutorium 14 -JavaDoc JavaDoc für Klassen

 beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist



- beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist
- mögliche Verwendungszwecke können erwähnt werden
- der Autor wird angegeben





 beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist

der Autor wird angegeben

JavaDoc für Klassen



beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist

JavaDoc

- mögliche Verwendungszwecke können erwähnt werden
- der Autor wird angegeben
- die Versionsnummer wird angegeben



Programmieren Tutorium

Java-API



 beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist

der Autor wird angegeben

die Versionsnummer wird angegeben

☐JavaDoc für Klassen



OO-Design-Prinzipien

beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist

JavaDoc

- mögliche Verwendungszwecke können erwähnt werden
- der Autor wird angegeben
- die Versionsnummer wird angegeben

Übungsblatt 4

Wiederholung

```
Syntax
/**
 * Kurze aber aussagekräftige Beschreibung der Klasse.
 * @author Autor
 * @version versionsNummer
class ClassName {
```

Exceptions 000 000000 00000 000000000 0000000000 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Java-API

Tutorium 14 -JavaDoc ☐JavaDoc für Klassen

 beschreibt, welchen Zweck die Klasse erfüllt und welche Funktionalität sie aufweist mögliche Verwendungszwecke können erwährt werden der Autor wird angegeben

die Versionsnummer wird angegeber

JavaDoc für Klassen - Beispiel



```
/**
  * Modelliert einen zweidimensionalen Vektor.
  * Der Vektor besteht aus einer x- und y-Komponente.
  * @author Peter Bohner
  * @author uqknc
  * @version 1.0
  */
class Vector2D {
    double x;
    double y;
}
```

```
Tutorium 14

JavaDoc

JavaDoc für Klassen - Beispiel

Gaster 900

Gaster 910

Gaster 910
```

Wiederholung

000

Übungsblatt 4

000000

JavaDoc

00000

Exceptions

000000000000000

OO-Design-Prinzipien

Java-API



Tutorium 14

JavaDoc

JavaDoc für Konstruktoren



beschreibt Zweck der Konstruktors

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API 000 000000 00000 000000000000000 000000000 0000000000

Tutorium 14 -JavaDoc ☐ JavaDoc für Konstruktoren

beschreibt Zweck der Konstruktors

Programmieren Tutorium

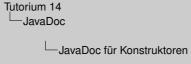
18.01.2023

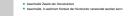
Péter Bohner: Tutorium 14

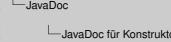
- beschreibt Zweck der Konstruktors
- beschreibt, in welchem Kontext der Kontruktor verwendet werden kann



Programmieren Tutorium







18.01.2023 Péter Bohner: Tutorium 14



- beschreibt Zweck der Konstruktors
- beschreibt, in welchem Kontext der Kontruktor verwendet werden kann
 - Spezialfälle, die das Verhalten betreffen



Tutorium 14 -JavaDoc ☐ JavaDoc für Konstruktoren

 beschreibt, in welchem Kontext der Kontruktor verwendet werden kann Spezialtälle, die das Verhalten betreffen

beschreibt Zweck der Konstruktors



- beschreibt Zweck der Konstruktors
- beschreibt, in welchem Kontext der Kontruktor verwendet werden kann

JavaDoc

00000

Exceptions

000000000000000

- Spezialfälle, die das Verhalten betreffen
- alle Parameter werden beschrieben (Zweck und eventuell Wertebereich)





Tutorium 14 -JavaDoc

☐ JavaDoc für Konstruktoren

 beschreibt Zweck der Konstruktors beschreibt, in welchem Kontext der Kontruktor verwendet werden kann

 Spezialtille, die das Verhalten betreffen alle Parameter werden beschrieben (Zweck und eventuell Wertebereich

Wiederholung

Übungsblatt 4



- beschreibt Zweck der Konstruktors
- beschreibt, in welchem Kontext der Kontruktor verwendet werden kann
 - Spezialfälle, die das Verhalten betreffen
- alle Parameter werden beschrieben (Zweck und eventuell Wertebereich)

```
Syntax
  * Kurze aber aussagekräftige Beschreibung von Zweck und Kontext des Konstruktors.
 * Sonderfälle werden beschrieben.
  * @param parameter Beschreibung des Parameters
KlassenName(Datentyp parameter, ...) {
```

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API 000000 00000 0000000000 000

Tutorium 14 -JavaDoc

☐ JavaDoc für Konstruktoren

 beschreibt, in welchem Kontext der Kontruktor verwendet werden kann Spezialtälle, die das Verhalten betreffen

beschreibt Zweck der Konstruktors

KlassenName(Datentyp parameter, ...) {

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

JavaDoc für Konstuktoren - Beispiel



```
/**
 * Konstruiert einen zwei dimensionalen Vektor mit zwei übergebenen Komponenten.
 * @param x die x-Komponente des Vektors
 * @param y die y-Komponente des Vektors
Vector2D(double x, double y) {
   this.x = x;
   this.y = y;
```

```
Übungsblatt 4
Wiederholung
                                               JavaDoc
                                                                                                  Java-API
                                                                                                                         OO-Design-Prinzipien
                                                                   Exceptions
000
                       000000
                                               0000
                                                                   000000000000000
                                                                                                  000000000
                                                                                                                         0000000000
      18.01.2023
                        Péter Bohner: Tutorium 14
```

Tutorium 14 -JavaDoc - gparae x die x-Komponente des Vektors JavaDoc für Konstuktoren - Beispiel Vector20(double x, double y) (

Exceptions - Einführung



Exception

- eine Ausnahme
- Zur Laufzeit des Programms
- Zur Unterbrechung des normalen Kontrollflusses

-Exceptions Exceptions - Einführung

Exception

eine Ausnahme Zur Laufzeit des Programms Zur Unterbrechung des normalen Kontrollflusses

Beispiele:

Tutorium 14

- 1. Eine Datei nicht gefunden wird.
- 2. Jemand durch 0 teilt.
- 3. Ungültige Argumente einer Methode übergeben wurden.
- 4. Versucht wird auf ein nicht-existentes Arrayelement zuzugreifen.

Wiederholung 000





18.01.2023 Péter Bohner: Tutorium 14

Programmieren Tutorium

Exceptions - Einführung



Exception

- eine Ausnahme
- Zur Laufzeit des Programms
- Zur Unterbrechung des normalen Kontrollflusses

Verwendung einer Exception

- Ein Problem tritt auf
- Normales Fortfahren nicht möglich
- Lokale Reaktion darauf nicht sinnvoll/möglich
- Behandlung des Problems an anderer Stelle nötig

 Tutorium 14
Exceptions
Exceptions - Einführung



Beispiele:

- 1. Eine Datei nicht gefunden wird.
- 2. Jemand durch 0 teilt.
- 3. Ungültige Argumente einer Methode übergeben wurden.
- 4. Versucht wird auf ein nicht-existentes Arrayelement zuzugreifen.



Exceptions in Java

Ausnahme in Java

- echtes Objekt (Methoden, Attribute, ...)
- Von Klasse Exception abgeleitet
- Mindestens zwei Konstruktoren: Default & mit String-Parameter (mit zusätzlichen Informationen)
- Methoden: getMessage() & printStackTrace()
- Erzeugung mit new
- Auslösen mit throw

Übungsblatt 4 JavaDoc Java-API OO-Design-Prinzipien Wiederholung Exceptions 000 000000 00000 000000000 0000000000 000000000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 -Exceptions -Exceptions

Exceptions in Java Ausnahme in Java echtes Objekt (Methoden, Attribute, . . Von Klasse Exception abgeleitet Mindestens zwei Konstruktoren: Default & mit Strung-Parameter (mit zusätzlichen Informationen)

 Erzeugung mit sew Auslösen mit throw

Exceptions - Beispiel



```
public void setMonth(int month) {
    if ((month < 1) || (month > 12)) {
        throw new IllegalArgumentException(
                "Wrong month: " + month);
    this.month = month;
```

```
Übungsblatt 4
Wiederholung
                                           JavaDoc
                                                             Exceptions
                                                                                          Java-API
000
                     000000
                                           00000
                                                             00000000000000
                                                                                          000000000
                                                                                                               0000000000
```

Tutorium 14 Exceptions Exceptions - Beispiel





Exceptions - Arten von Fehlern



Error

(Katastrophale) Probleme, die eigentlich nicht auftreten dürfen. Speicher voll, Illegaler Byte-Code, JVM-Fehler, ...

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000000 00000 000000000000000 000000000 0000000000

Tutorium 14 -Exceptions Exceptions - Arten von Fehlern

(Katastrophale) Probleme, die eigentlich nicht auftreten dürfen. Speicher voll, lliegaler Byte-Code, JVM-Fehler, . . .

Exceptions - Arten von Fehlern



OO-Design-Prinzipien

Error

Wiederholung

(Katastrophale) Probleme, die eigentlich nicht auftreten dürfen.

JavaDoc

Speicher voll, Illegaler Byte-Code, JVM-Fehler, ...

RuntimeException

Durch fremde Fehler erzeugte Probleme.

falsche Benutzung einer Klasse, Programmierfehler

Übungsblatt 4

Exceptions 000 000000 00000 000000000000000 000000000 0000000000 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 -Exceptions Exceptions - Arten von Fehlern

(Katastrophale) Probleme, die eigentlich nicht auftreten dürfen. Speicher voll, lilegaler Byte-Code, JVM-Fehler, ...

Durch fremde Fehler erzeugte Probleme. falsche Benutzung einer Klasse, Programmierfehler

Exceptions - Arten von Fehlern



Tutorium 14 -Exceptions Exceptions - Arten von Fehlern

(Katastrophale) Probleme, die eigentlich nicht auftreten dürfen. Speicher voll, lilegaler Byte-Code, JVM-Fehler, ... Durch frende Fehler erzeugte Probleme. falsche Benutzung einer Klasse, Programmierfehler Datei nicht vorhanden. Festplatte voll. Fehler beim Parsen.

Error

(Katastrophale) Probleme, die eigentlich nicht auftreten dürfen. Speicher voll, Illegaler Byte-Code, JVM-Fehler, ...

RuntimeException

Durch fremde Fehler erzeugte Probleme. falsche Benutzung einer Klasse, Programmierfehler

Geprüfte Exception (checked Exception)

Vorhersehbare und behandelbare Fehler.

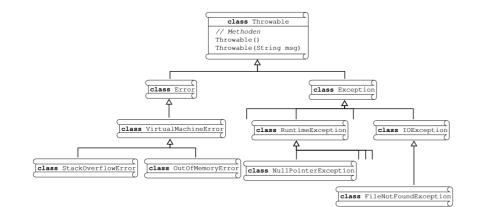
Datei nicht vorhanden, Festplatte voll, Fehler beim Parsen, ...

Übungsblatt 4 Java-API OO-Design-Prinzipien Wiederholung JavaDoc Exceptions 000000 00000 000000000 0000000000 000000000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Exceptions - Hierarchie







21/52 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14
Exceptions

Exceptions - Hierarchie



18.01.2023

Péter Bohner: Tutorium 14



```
Ausnahmebehandlung in Java

try {
    // hier koennte eine Exception auftreten
} catch (ExceptionType1 e) {
    // Fehlerbehandlung fuer ExceptionType1
} catch (ExceptionType2 e) {
    // Fehlerbehandlung fuer ExceptionType2
```



Programmieren Tutorium

Tutorium 14

Exceptions

Exceptions



```
Ausnahmebehandlung in Java
try {
    // hier koennte eine Exception auftreten
  catch (ExceptionType1 e) {
    // Fehlerbehandlung fuer ExceptionType1
  catch (ExceptionType2 e) {
    // Fehlerbehandlung fuer ExceptionType2
```

Fall-through, die Zweite

- Java ruft den ersten passenden catch Block auf!
- Alle weiteren werden ignoriert

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions 000 000000 00000 000000000 0000000000 000000000000000

Tutorium 14 -Exceptions -Exceptions

// hier kommitte eine Exception auftrete } catch (ExceptionTypel e) { } catch (ExceptionType2 e) { Alle weiteren werden ignorier?



```
Beispiel
try {
    FileReader fr = new FileReader(".test");
    int nextChar = fr.read();
    while (nextChar != -1) {
        nextChar = fr.read();
} catch (FileNotFoundException e) {
    System.out.println("Nicht gefunden.");
} catch (IOException e) {
    System.out.println("Ooops.");
```

```
Übungsblatt 4
Wiederholung
                                             JavaDoc
                                                                                              Java-API
                                                                                                                    OO-Design-Prinzipien
                                                                Exceptions
000
                      000000
                                            00000
                                                                000000000000000
                                                                                              000000000
                                                                                                                    0000000000
```

Tutorium 14 Exceptions Exceptions

FileReader fr = new FileReader(".test"); int nextChar = fr.read(); while (nextChar != -1) { nextChar = fr.read(); } catch (FileNotFoundException e) { System.out.println('Nicht gefunden.') } catch (IOException e) { System.out.println("Goops.");



Exception Handler (= catch-Block)

- Behandlung einer Ausnahme
- an einer Stelle
- irgendwo im Aufrufstack
- getrennt von normalen Programmcode

OO-Design-Prinzipien 0000000000

Übungsblatt 4

Péter Bohner: Tutorium 14

000000

JavaDoc

00000

Exceptions

000000000000000

Java-API

000000000

-Exceptions Exceptions

Tutorium 14

Exception Handler (= catch-Block • Behandlung einer Ausnahme iroendwo im Aufrufstack getrennt von normalen Programmcode



Exception Handler (= catch-Block)

- Behandlung einer Ausnahme
- an einer Stelle irgendwo im Aufrufstack
- getrennt von normalen Programmcode

Catch or specify

Jede ausgelöste geprüfte (checked) Exception muss

- behandelt (Exception Handler) oder

deklariert (throws)										
werden.										
Wiederholung	Übungsblatt 4	JavaDoc 00000	Exceptions	Java-API	OO-Design-Prinzipien					

Tutorium 14 -Exceptions Exceptions





Deklaration von Ausnahmen

Deklaration im Methodenkopf: private String readFile(String filename) throws IOException, FileNotFoundException {}

 Wiederholung
 Übungsblatt 4
 JavaDoc
 Exceptions
 Java-API
 OO-Design-Prinzipien

 000
 000000
 0000000
 0000000
 00000000
 00000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14

Exceptions

Exceptions

Objection to Associate

Solution is Manufacture

Solution is Manufacture

priorite String restficitizing fileman) three Indicaption, filestfundicaption ()



Deklaration von Ausnahmen

- Deklaration im Methodenkopf: private String readFile(String filename) throws IOException, FileNotFoundException {}
- Aufrufer muss sich um Exception kümmern
- throws ist Teil der Signatur (Vorsicht beim Überschreiben) Exceptions können in überschriebenen Methoden weggelassen werden, aber nicht hinzukommen.
- Nicht deklarationspflichtig sind RuntimeException & Error (sowie deren Unterklassen)
- Jede Exception mittels @throws im Javadoc beschrieben werden



-Exceptions

Tutorium 14

grivate String resifile(String filename) throws IOException. FileNotFoundException (Aufrufer muss sich um Exception kümmen

Nicht deklarationspflichtig sind Runt ineException & Error (sowie deren Unterklassen)

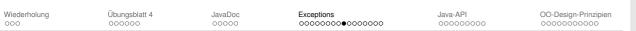
Jede Exception mittels Øthrows im Javadoc beschrieben werden



Deklaration von Ausnahmen

- Deklaration im Methodenkopf: private String readFile(String filename) throws IOException, FileNotFoundException {}
- Aufrufer muss sich um Exception kümmern
- **throws** ist Teil der Signatur (Vorsicht beim Überschreiben) Exceptions können in überschriebenen Methoden weggelassen werden, aber nicht hinzukommen.
- Nicht deklarationspflichtig sind RuntimeException & Error (sowie deren Unterklassen)
- Jede Exception mittels @throws im Javadoc beschrieben werden

Anmerkung: Da 10Exception Oberklasse von FileNotFoundException ist, müsste letzteres nicht extra deklariert werden. Dokumentationszwecke!



Tutorium 14 Exceptions Exceptions

grivate String resifile(String filename) throws IOException. FileNotFoundException (Aufrufer muss sich um Exception kümmen throws ist Teil der Signatur (Vorsicht beim Überschreiben)

Nicht deklarationspflichtig sind Runt ineException & Error (sowie deren Unterklassen)

Anmerkung: Da 10Except son Oberklasse von Fs Lellot FoundExcept son ist, müsste letzteres nicht ext.

doktoried warden. Onkumentations sweeket

18.01.2023



Ort der Behandlung

Finden der passenden Ausnahmebehandlung:

Péter Bohner: Tutorium 14

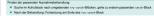
- Suche im Aufrufstack nach umgebenden try-catch-Blöcken, gehe zu erstem passenden catch-Block
- Nach der Behandlung: Fortsetzung am Ende des try-catch-Block

 Wiederholung
 Übungsblatt 4
 JavaDoc
 Exceptions
 Java-API
 OO-Design-Prinzipien

 000
 00000
 00000
 00000000
 00000000
 00000000

Programmieren Tutorium









Behandlung?

- Error und Unterklassen: Nein, nicht sinnvoll behandelbar
- **Exception:** Nein, viel zu allgemein
- RuntimeException: Prinzipiell Nein
- dessen Unterklassen: Programmierfehler beheben! (Ausnahme: NumberFormatException)
- Andere: Ja, wenn sinnvoll behandelbar
- try-Block so klein wie möglich halten



7/52 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14
Exceptions
Exceptions - Konventionen

Error und Unterklassen: Nein, nicht sinnvoll behandelber

Exception: Nein, viel zu allgemein
 RuntimeException: Prinzipiell Nein

dessen Unterklassen: Programmierlehler beheben! (Ausnahme: NumberFormstException)

Andere: Ja, wern sinnvoll behandelbar
 try-Block so klein wie möglich halten



Werfen?

- Error: Nein.
- Exception: Niemals, nur als eigene Unterklasse
- **RuntimeException:** Ja, eigene (semantisch passende) Unterklasse

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000000 00000 00000000000000000 000000000 0000000000





18.01.2023

000

Péter Bohner: Tutorium 14

Programmieren Tutorium



Werfen?

- Error: Nein.
- **Exception:** Niemals, nur als eigene Unterklasse
- RuntimeException: Ja, eigene (semantisch passende) Unterklasse

```
Beispiel
if ((month < 1) || (month > 12)) {
    throw new IllegalArgumentException(
             "Wrong month: %s", month);
switch (month) {
    case 1: break; // ...
    default: throw new Error();
               Übungsblatt 4
                                 JavaDoc
                                                Exceptions
                                                                                        OO-Design-Prinzipier
                000000
                                 00000
                                                                       000000000
                                                                                        0000000000
```

Tutorium 14

Exceptions

Exceptions - Konventionen

Exceptions - Konventionen

Exceptions - Konventionen



Verwendung

000

18.01.2023

Exceptions sollen:

- zur Vereinfachung dienen
- die absolute Ausnahme darstellen
- mittels @throws im Javadoc beschrieben werden

Péter Bohner: Tutorium 14

NICHT den normalen Kontrollfluss steuern

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000000 00000 000000000 0000000000

Programmieren Tutorium







Verwendung

Exceptions sollen:

- zur Vereinfachung dienen
- die absolute Ausnahme darstellen
- mittels @throws im Javadoc beschrieben werden
- NICHT den normalen Kontrollfluss steuern

```
Böse!

try {
    while (character != array[i]) { i++; }
} catch (Exception e) {
    System.out.println("Element nicht gefunden.");
}
```

Exceptions

Java-API

000000000

Tutorium 14
Exceptions
Exceptions - Konventionen



Wiederholung

JavaDoc

00000

Übungsblatt 4

000000

OO-Design-Prinzipien

0000000000



0000000000

Verboten!

000

- try-Block um das ganze Programm
- Leerer catch-Block
- Explizites Fangen des Typs Exception
- Explizites Fangen des Typs Throwable

000000

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien

0000000000000000

000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

00000

Tutorium 14 -Exceptions Exceptions - Konventionen

try-Block um das ganze Programm

 Leerer catch-Block Explizites Fangen des Typs Exception

Explizites Fangen des Typs Throusble



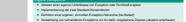
Eigene Exceptions

- Ableiten einer eigenen Unterklasse von Exception oder RuntimeException
- Implementierung der zwei Standard-Konstruktoren
- Definition einer eigenen, sinnvollen Exception-Hierarchie (bei Bedarf)
- Verwendung von vorhandenen Exceptions nur für dafür vorgesehene Zwecke (Javadoc anschauen)



/52 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Tutorium 14
Exceptions
Exceptions





Eigene Exceptions

- Ableiten einer eigenen Unterklasse von Exception oder RuntimeException
- Implementierung der zwei Standard-Konstruktoren
- Definition einer eigenen, sinnvollen Exception-Hierarchie (bei Bedarf)
- Verwendung von vorhandenen Exceptions nur f
 ür daf
 ür vorgesehene Zwecke (Javadoc anschauen)

Beispiele in der Java-API

- IllegalArgumentException
- IllegalStateException
- UnsupportedOperationException
- NullPointerException

Wiederholung	Übungsblatt 4	JavaDoc	Exceptions	Java-API	OO-Design-Prinzipien
000	000000	00000	000000000000000	00000000	0000000000

Tutorium 14 -Exceptions -Exceptions

- Definition einer eigenen, sinnvollen Exception-Hierarchie (bei Bedarf)
- Verwendung von vorhandenen Exceptions nur f
 ür daf
 ür vorsesehene Zwecke (Javadoc anschauen)
 - IllegalArgumentException
 - IllegalStateException
 - UnsupportedOperationException
 - NullPointerException

Exceptions - Zusammenfassung



Ausnahmen

- werden ausgelöst (throw) und behandelt (try-catch) oder
- deklariert (throws)
- sollen die Ausnahme bleiben
- trennen sauber Programmlogik und Fehlerbehandlung

-Exceptions

Tutorium 14

Exceptions - Zusammenfassung

 werden ausgelöst (throw) und behandelt (try-catch) oder deklariert (throws) sollen die Ausnahme bleiben trennen sauber Programmlogik und Fehlerbehandlung

Exceptions - Zusammenfassung



Ausnahmen

- werden ausgelöst (throw) und behandelt (try-catch) oder
- deklariert (throws)
- sollen die Ausnahme bleiben
- trennen sauber Programmlogik und Fehlerbehandlung

Fehlererkennung

- so früh wie möglich
- defensiv
- mittels if Exceptions

Übungsblatt 4 JavaDoc Java-API OO-Design-Prinzipien Exceptions 000000 00000 000000000000000 000000000 0000000000

Tutorium 14 -Exceptions Exceptions - Zusammenfassung

 werden ausgelöst (throw) und behandelt (try-catch) oder deklariert (throws) sollen die Ausnahme bleiben trennen sauber Programmlogik und Fehlerbehandlung so früh wie möglich mittels if Exceptions

Wiederholung 000

18.01.2023

Péter Bohner: Tutorium 14

Programmieren Tutorium

Java-API - Application Programming Interface

- Sammlung von Klassen und Paketen
- Enthalten häufig verwendete Funktionalität



Tutorium 14 -Java-API

Java-API - Application Programming Interface

 Sammlung von Klassen und Paketen Enthalten häufig verwendete Funktionalität

1. Klassen in java.lang: Object, String, Math, Enum, ...

Wiederholung

Übungsblatt 4 000000

JavaDoc 00000

Exceptions 000000000000000

Java-API

OO-Design-Prinzipien 0000000000

•00000000 Péter Bohner: Tutorium 14

Programmieren Tutorium

Java-API - Application Programming Interface



- Sammlung von Klassen und Paketen
- Enthalten häufig verwendete Funktionalität

Bekannte Klassen

- Object
- String, Math, Enum
- Comparable, Integer, Double, ...



Tutorium 14

Java-API

___Java-API - Application Programming Interface

Ershalten häufig vernendene Funktionaltät
Bellamnink Klasson
 Opiect
 String Rath, Etus
 Cosparalte, Zeteper, Doable, ...

Sammlung von Klassen und Paketen

1. Klassen in java.lang: Object, String, Math, Enum, ...

Wiederholung 000

Péter Rohner: Tutorium

Übungsblatt 4

000000

JavaDoc 00000

Java-API

Programmieren Tutorium

OO-Design-Prinzipien

0000000000

//52 18.01.2023 Péter Bohner: Tutorium 14 Programmie

Exceptions

Java-API - Application Programming Interface



- Sammlung von Klassen und Paketen
- Enthalten häufig verwendete Funktionalität

Bekannte Klassen

- Object
- String, Math, Enum
- Comparable, Integer, Double, ...

Pakete java.lang Basisfunktionen (oben genannte Klassen) java.util Java Collections, Zeit-/Datumsfunktionen java.io Ein-/Ausgabe Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions 000000 00000 •00000000 0000000000 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium -Java-API Java-API - Application Programming Interface

Tutorium 14

 Enthalten häufig verwendete Funktionalität • String, Math, Enum · Comparable, Integer, Double, java. Long Basisfunktionen (oben genannte Klassen) java.util Java Collections, Zeit-/Datumsfunktionen java. to Ein-/Ausgabe

Sammlung von Klassen und Paketen

1. Klassen in java.lang: Object, String, Math, Enum, ...

Java-API - Collection



Das Interface java.util.Collection<E>

- java.util.List
- java.util.Set
- java.util.SortedSet
- java.util.Queue

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API 000 000000 00000000

☐Java-API - Collection

Tutorium 14 -Java-API



- 1. Die unten stehenden Interfaces erben vom Collection-Interface
- 2. List -> Liste mit implementierenden ArrayList, LinkedList, Stack...
- 3. Set -> Menge mit implementierenden Klassen HashSet, TreeSet, ...
- 4. SortedSet -> sortierte Menge (nach compareTo wird sortiert) mit implementierenden Klassen TreeSet
- 5. Queues -> Warteschlangen, PriorityQueue, sind aber absolut unwichtig
- 6. HashSet -> Elemente werden gehasht (int-Wert bestimmen) und in einen Eimer mit demselben Hash gepackt, in diesem Eimer wird nochmal geprüft, ob Objekt schon enthalten ist (equals-Methode)
- 7. TreeSet -> Elemente werden in einem binären-Suchbaum gespeichert (Rot-Schwarz-Baum, ist egal was das ist), damit man leicht auf einzelne Elemente zugreifen kann (Suchen kann), und nicht durch den gesamten Baum iterieren muss

Java-API - Collection



Das Interface java.util.Collection<E>

- java.util.List
- java.util.Set
- java.util.SortedSet
- java.util.Queue

Wichtige Methoden

- boolean contains(Object element)
- boolean add(E element)
- boolean remove(Object element)

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API 000 000000 00000 00000000

☐Java-API - Collection

Tutorium 14 Java-API



- 1. Die unten stehenden Interfaces erben vom Collection-Interface
- 2. List -> Liste mit implementierenden ArrayList, LinkedList, Stack...
- 3. Set -> Menge mit implementierenden Klassen HashSet, TreeSet, ...
- 4. SortedSet -> sortierte Menge (nach compareTo wird sortiert) mit implementierenden Klassen TreeSet
- 5. Queues -> Warteschlangen, PriorityQueue, sind aber absolut unwichtig
- 6. HashSet -> Elemente werden gehasht (int-Wert bestimmen) und in einen Eimer mit demselben Hash gepackt, in diesem Eimer wird nochmal geprüft, ob Objekt schon enthalten ist (equals-Methode)
- 7. TreeSet -> Elemente werden in einem binären-Suchbaum gespeichert (Rot-Schwarz-Baum, ist egal was das ist), damit man leicht auf einzelne Elemente zugreifen kann (Suchen kann), und nicht durch den gesamten Baum iterieren muss

Java-API - Map



Das Interface java.util.Map<K, V>

- Paare werden eindeutig nach ihrem Key gespeichert
- Ein Key darf nur einmal enthalten sein

Tutorium 14

— Java-API

— Java-API - Map



- 1. Datenstruktur, die einem Schlüssel und Werte-Paare speichert -> Schlüssel darf nur einmal in der Menge enthalten sein; Funktion, die von K nach V abbildet/mapt
- 2. put: the previous value associated with key, or null if there was no mapping for key.
- 3. Es gibt die Implementierungen HashMap (keys werden gehasht) und TreeMap (nach Keys sortierte Speicherung im Baum nach der compareTo-Methode)

Java-API - Map



Das Interface java.util.Map<K, V>

- Paare werden eindeutig nach ihrem Key gespeichert
- Ein Key darf nur einmal enthalten sein

Wichtige Methoden

- V put(K key, V value)
- V get(Object key)
- V remove(Object key)
- boolean containsKey(Object key)
- boolean containsValue(Object value)

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions Java-API 0000000000 000 000000 00000000

Tutorium 14 -Java-API

└─Java-API - Map



- 1. Datenstruktur, die einem Schlüssel und Werte-Paare speichert -> Schlüssel darf nur einmal in der Menge enthalten sein; Funktion, die von K nach V abbildet/mapt
- 2. put: the previous value associated with key, or null if there was no mapping for key.
- 3. Es gibt die Implementierungen HashMap (keys werden gehasht) und TreeMap (nach Keys sortierte Speicherung im Baum nach der compareTo-Methode)

Java-API - Map



```
Beispiel
```

Wiederholung

18.01.2023

000

```
public static void main(String[] args) {
   Map<Integer, Integer> intMap = new HashMap<Integer, Integer>();
   intMap.put(5, 20);
   intMap.put(10, 30);
   if (intMap.containsKey(5)) {
       System.out.println(intMap.get(5)); // 20
```

JavaDoc

00000

Exceptions

000000000000000

Übungsblatt 4

Péter Bohner: Tutorium 14

000000

OO-Design-Prinzipien

Programmieren Tutorium

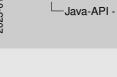
0000000000

Java-API

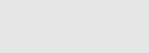
000000000

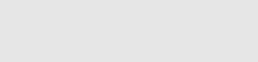
```
└─Java-API
     └─Java-API - Map
```





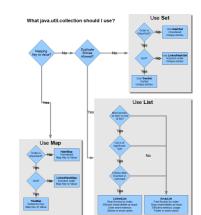
Tutorium 14





Java-API - Welche Collection nehme ich wann?





Quelle: https://i.stack.imgur.com/aSDsG.png

Übungsblatt 4 Wiederholung JavaDoc Exceptions Java-API OO-Design-Prinzipien 000000 00000 000000000000000 000000000 0000000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 -Java-API Java-API - Welche Collection nehme ich wann?



Quelle Milya: //L. stant. impo. com/all/sl. pro

Java-API - Reguläre Ausdrücke



java.util.regex und seine Klassen

- Pattern (java.util.regex.Pattern)
- Matcher (java.util.regex.Matcher)



Tutorium 14 -Java-API

☐ Java-API - Reguläre Ausdrücke

Pattern (java.util.regex.Pattern) Matcher (java.util.regex.Matcher)

1. Backslash zum escapen des Backslashes, + ist selbst Quantor und muss deshalb auch mit Backslash angegeben

werden

Wiederholung 000

00000

JavaDoc

Exceptions

000000000000000

Java-API 000000000 OO-Design-Prinzipien

0000000000

18.01.2023

Übungsblatt 4

000000

Péter Bohner: Tutorium 14

Programmieren Tutorium

Java-API - Reguläre Ausdrücke



```
java.util.regex und seine Klassen
```

- Pattern (java.util.regex.Pattern)
- Matcher (java.util.regex.Matcher)

```
String positiveIntegerPattern = "\\+?[1-9]\\d*";
Pattern pattern = Pattern.compile(positiveIntegerPattern);
Matcher matcher = pattern.matcher("+529");
boolean isMatch = matcher.matches(); // true
```

JavaDoc

00000

Exceptions





Pattern pattern = Pattern.compile(positiveIntegerPattern); Matcher matcher = pattern.matcher("+529"); boolean isMatch = matcher.matches(); // true

Pattern (java.util.regex.Pattern) Matcher (java.util.regex.Matcher)

String positiveIntegerPattern = "\\+7[1-9]\\d+";

1. Backslash zum escapen des Backslashes, + ist selbst Quantor und muss deshalb auch mit Backslash angegeben werden

000

Übungsblatt 4

OO-Design-Prinzipien

0000000000

Java-API

000000000

Java-API - Reguläre Ausdrücke



Reguläre Ausdrücke werden wie in GBI gebildet

Character

- Klassen
 - \d Ziffer
 - \w Zeichen
 - [a-q] Ein Zeichen von a-g, [1-9] Ziffer von 1-9
- Quantoren

 - * Ausdruck davor 0 mal oder mehr, + Ausdruck davor mindestens 1 mal
 - ? Ausdruck davor 0 oder 1 mal
 - a{5} 5 mal a

Reguläre Ausdrücke testen: https://regexr.com/

Übungsblatt 4 Java-API OO-Design-Prinzipien Wiederholung JavaDoc Exceptions 000000 00000 000000000 0000000000

Tutorium 14 -Java-API Java-API - Reguläre Ausdrücke



1. Dazu sagen, dass in Java mehrere Backslashs nötig sind, um beispielsweise das d oder das + dort unterzubringen



Pattern Schreibe jeweils für die folgenden Muster einen regulären Ausdruck, sodass diese mit Hilfe der Klasse Pattern von Java erkannt werden würden.

-Java-API └─Aufgabe

Tutorium 14

1. Capture Groups eventuell im Kopf haben bei Fragen

Wiederholung

Exceptions

000000000000000

OO-Design-Prinzipien 0000000000

000

18.01.2023

000000 Péter Bohner: Tutorium 14

Übungsblatt 4

JavaDoc 00000

Java-API 000000000

Programmieren Tutorium



OO-Design-Prinzipien

Programmieren Tutorium

0000000000

Tutorium 14 2023-01-18

-Java-API

1. Capture Groups eventuell im Kopf haben bei Fragen

5-stellige positive Ganzzahl (ohne +)

5-stellige positive Ganzzahl (ohne +)





18.01.2023

Übungsblatt 4 000000

Péter Bohner: Tutorium 14

JavaDoc 00000

Exceptions

000000000000000





Java-API

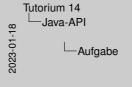


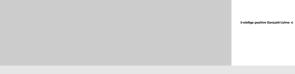
















Wiederholung

18.01.2023

000



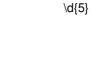






JavaDoc

00000



Exceptions

000000000000000





Java-API

000000000





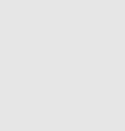
OO-Design-Prinzipien

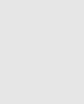
Programmieren Tutorium

0000000000



1. Capture Groups eventuell im Kopf haben bei Fragen







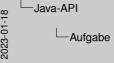


Péter Bohner: Tutorium 14









Tutorium 14

5-stellige positive Ganzzahl (ohne +) Ganzzahl (mit + und -)

Wiederholung

18.01.2023

000

5-stellige positive Ganzzahl (ohne +)

Ganzzahl (mit + und -)

Übungsblatt 4

Péter Bohner: Tutorium 14

000000

JavaDoc

00000





000000000000000

Exceptions

 $\d{5}$







Java-API

000000000













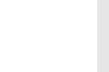






OO-Design-Prinzipien

Programmieren Tutorium











1. Capture Groups eventuell im Kopf haben bei Fragen













Tutorium 14

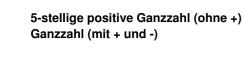
5-stellige positive Ganzzahl (ohne +) Ganzzahl (mit + und -)

\d(5) (+|-)?\d*









Wiederholung

18.01.2023

000



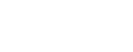


Péter Bohner: Tutorium 14

JavaDoc

00000





000000000000000

Exceptions

 $\d{5}$

(+|-)?\d*





Java-API

000000000







OO-Design-Prinzipien

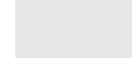
Programmieren Tutorium

0000000000









1. Capture Groups eventuell im Kopf haben bei Fragen



└─Aufgabe

Tutorium 14

-Java-API

5-stellige positive Ganzzahl (ohne +) Ganzzahl (mit + und -)

1. Capture Groups eventuell im Kopf haben bei Fragen

5-stellige positive Ganzzahl (ohne +) Ganzzahl (mit + und -) kleingeschriebenes Wort, anschließend Doppelpunkt

 $\d{5}$ $(+|-)?\d^*$

OO-Design-Prinzipien

Wiederholung 000

Übungsblatt 4 000000

Exceptions 000000000000000

Java-API

0000000000

18.01.2023

Péter Bohner: Tutorium 14

JavaDoc 00000

000000000 Programmieren Tutorium



└─Aufgabe

Tutorium 14 -Java-API

5-stellige positive Ganzzahl (ohne +)

Ganzzahl (mit + und -)

1. Capture Groups eventuell im Kopf haben bei Fragen

5-stellige positive Ganzzahl (ohne +) Ganzzahl (mit + und -) kleingeschriebenes Wort, anschließend Doppelpunkt

 $\d{5}$ (+|-)?\d*

OO-Design-Prinzipien Java-API 000000000 0000000000

Wiederholung 000

Übungsblatt 4 000000

Péter Bohner: Tutorium 14

JavaDoc 00000

Exceptions 000000000000000

Programmieren Tutorium

18.01.2023



└─Aufgabe

Tutorium 14 -Java-API

> 5-stellige positive Ganzzahl (ohne +) Ganzzahl (mit + und -) kleingeschriebenes Wort, anschließend Doppelpunkt | w+: positive Kommazahl (mit + und -)

1. Capture Groups eventuell im Kopf haben bei Fragen

5-stellige positive Ganzzahl (ohne +) Ganzzahl (mit + und -) kleingeschriebenes Wort, anschließend Doppelpunkt

positive Kommazahl (mit + und -)

 $\d{5}$ (+|-)?\d*

Exceptions

Wiederholung

000000000000000

Java-API

000000000

OO-Design-Prinzipien

000

18.01.2023

000000 Péter Bohner: Tutorium 14

Übungsblatt 4

JavaDoc 00000

0000000000

Programmieren Tutorium



└─Aufgabe

Tutorium 14 -Java-API

> 5-stellige positive Ganzzahl (ohne +) positive Kommazahl (mit + und -)

1. Capture Groups eventuell im Kopf haben bei Fragen

5-stellige positive Ganzzahl (ohne +) Ganzzahl (mit + und -)

positive Kommazahl (mit + und -)

Übungsblatt 4

 $\d{5}$ (+|-)?\d* kleingeschriebenes Wort, anschließend Doppelpunkt

Exceptions

 $(+|-)?\d+(,\d+)?$

JavaDoc



Java-API





Java-API - Wichtige Links



Allgemein

https://docs.oracle.com/en/java/javase/11/docs/api/index.html

java.util.Collection<E>

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Collection.html

java.util.Map<K, V>

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Map.html

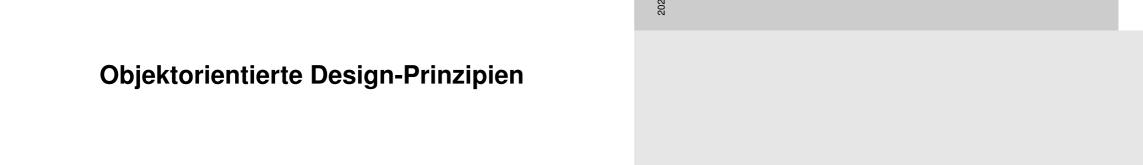
java.util.regex

https://docs.oracle.com/en/java/javase/11/docs/api/ java.base/java/util/regex/package-summary.html

Übungsblatt 4 Java-API OO-Design-Prinzipien Wiederholung JavaDoc Exceptions 000000 0000000000 000 00000000

18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Tutorium 14 -Java-API ☐ Java-API - Wichtige Links

- https://docs.oracle.com/en/lava/lavase/11/docs/agi/index.html • fave.util.CollectionsD>
- https://docs.oracle.com/en/laya/layase/11/docs/agi/laya.base/laya/util/Collection.html
- https://docs.oracle.com/en/lava/lavane/11/docs/agi/lava.base/lava/util/Map.html
- https://docs.oracle.com/en/java/javase/11/docs/agi/ java.base/java/util/regex/package-summary.html





Tutorium 14

OO-Design-Prinzipien

Übungsblatt 4 JavaDoc Exceptions OO-Design-Prinzipien 00000 000000000000000 000000000 •0000000000

Wiederholung 18.01.2023

000000 Péter Bohner: Tutorium 14

Programmieren Tutorium

Objektorientierte Design-Prinzipien

OO-Design-Prinzipien



- Datenkapselung
- Komposition vor Vererbung
- Gegen Schnittstellen programmieren

Übungsblatt 4

SOLID-Prinzipien

Wiederholung

Guter Schnittstellenentwurf

0000000000

OO-Design-Prinzipien

JavaDoc 000 000000 00000 000000000000000 000000000 18.01.2023 Péter Bohner: Tutorium 14

Exceptions

Java-API

Programmieren Tutorium

Tutorium 14 OO-Design-Prinzipien 2023-01-18

OO-Design-Prinzipien

 Datenkapselung Komposition vor Vererbung Gegen Schnittstellen programmieren a souze-Prinzipien

Guter Schnittstellenentwurf

Datenkapselung



- private Attribute
- Entwurfsentscheidung hinter Schnittstelle verdecken
- Keine Objektattribute in gettern direkt zurückgeben

Kopie von Collection erstellen und diese zurückgeben

JavaDoc

00000

Exceptions

000000000000000



OO-Design-Prinzipien

Java-API

000000000

Programmieren Tutorium



□ Datenkapselung

private Attribute

Keine Objektattribute in gettern direkt zurückgeben

Kopie von Collection erstellen und diese zunückgeben

18.01.2023

Wiederholung

000

Péter Bohner: Tutorium 14

Übungsblatt 4

Komposition vor Vererbung



- Komposition
 - Verwendung von Klassen als Attributtypen
 - Bestehende Objekte werden zusammengesetzt, um Funktionalität zu erweitern

JavaDoc

00000

Exceptions

000000000000000

Java-API

000000000



Programmieren Tutorium

OO-Design-Prinzipien

0000000000

Tutorium 14

OO-Design-Prinzipien

Komposition vor Vererbung

Komposition

Verwendung von Klassen als Attributivoen

Wiederholung 18.01.2023

000

Péter Bohner: Tutorium 14

Übungsblatt 4

Komposition vor Vererbung



- Komposition
 - Verwendung von Klassen als Attributtypen
 - Bestehende Objekte werden zusammengesetzt, um Funktionalität zu erweitern
- Vererbung
 - Ableitung von Eigenschaften und Funktionalität
 - Implementierung wird erweitert



Programmieren Tutorium



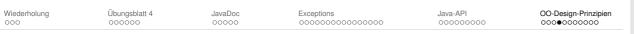


Ableitung von Eigenschaften und Funktionalität
 Implementierung wird erweitert

Komposition vor Vererbung



- Komposition
 - Verwendung von Klassen als Attributtypen
 - Bestehende Objekte werden zusammengesetzt, um Funktionalität zu erweitern
- Vererbung
 - Ableitung von Eigenschaften und Funktionalität
 - Implementierung wird erweitert
- Komposition sollte vor Vererbung genutzt werden, insofern das möglich ist
- Beispiel Klasse AccessControl
 - Attribut vom Typ Map zur Nutzerverwaltung
 - Man könnte auch AccessControl von HashMap ableiten und somit die Funktionalität zur Verfügung stellen
 - Ist aber unübersichtlich und verkompliziert den Code
 - Verletzt außerdem die is-a-Beziehung: AccessControl ist keine HashMap



Tutorium 14

OO-Design-Prinzipien

OO-Design-Filizipien

—Komposition vor Vererbung

Komposition
 Verwendung von Klassen als Attributypen

Verwendung von Klassen als Attributypen
 Bastehende Objekte werden zusammengesetzt, um Funktionalität zu erweiter

Ableitung von Eigenschaften und Funktionalität
 Implementierung wird erweitert

Komposition solite vor Vererbung genutzt werden, insofern das möglich ist
 Beispiel Klasse AccessControl.

Attribut som Typ Hup zur Nutzerverweitung

 Man könnte auch Accessiont roll von Hashitisp ableiten und somit die Funktionalität zur Verfügung stellen Ist aber unübersichtlich und verkompliziert den Code

Verletzt außerdem die is-a-Beziehung: AccessControl let keine HashMa

Gegen Schnittstellen programmieren



Nicht gegen die Implementierung programmieren!

- Leichte Austauschbarkeit
- Bessere Wiederverwendbarkeit

```
public int getStretchedCollectionSize(Collection<?> collection) {
    return collection.size() * this.factor;
```

JavaDoc

00000

Exceptions

```
OO-Design-Prinzipien
 0000000000
Programmieren Tutorium
```

000000000

Tutorium 14 OO-Design-Prinzipien

Gegen Schnittstellen programmieren

 Leichte Austauschbarkeit Bessere Wiederverwendbarkeit

return collection.size() - this.factor:

Nicht gegen die Implementierung programmieren!

Wiederholung 000

18.01.2023

Péter Bohner: Tutorium 14

Übungsblatt 4

SOLID-Prinzipien



- Jede Klasse sollte eine Verantwortung haben
- Open-Close Prinzip (n\u00e4chste Folie)
- Eine Instanz einer Unterklasse sollte sich so verhalten, dass sie jederzeit auch als Instanz der Oberklasse verwendet werden könnte
- Klassen sollten durch ein Interface nicht unnötige Methoden implementieren müssen ⇒ Interface aufteilen
- Abstraktes darf nicht von Details abhängen







Eine Instanz einer Unterklasse sollte sich so verhalten, dass sie jederzeit auch als Instanz der Oberklasse verwendet werden könnte Klassen sollten durch ein Interlace nicht unnötige Methoden implementieren müssen → Interlace aufteilen

Abstroktes riarf nicht von Datoils abhännen

Abstraktes darf nicht von Details abhängen

JavaDoc

00000



```
Schlecht
public class UserDatabase {
    LinkedList<User> users;
    public UserDatabase() {
        this.users = new LinkedList<>();
```

Übungsblatt 4

000000

Wiederholung

000



```
Tutorium 14
OO-Design-Prinzipien
```

Abstraktes darf nicht von Details abhängen

public class UserDatabase (LinkedList<Dser> users;

1. Code muss nur an einer Stelle geändert werden, wenn man die Datenstruktur ändern möchte (LinkedList zu ArrayList bspw.)

```
OO-Design-Prinzipien
00000000000
```

Java-API

Abstraktes darf nicht von Details abhängen



Tutorium 14

```
Schlecht
public class UserDatabase {
   LinkedList<User> users;

public UserDatabase() {
    this.users = new LinkedList<>();
   }
}

Gut
public class UserDatabase {
   List<User> users;

public UserDatabase() {
   this.users = new LinkedList<>();
   }
}
```

```
— OO-Design-Prinzipien

Abstraktes darf nicht von Details abhängen

1. Code muss nur an einer Stelle geändert werden, wenn man die Datenstruktur ändern möchte (LinkedList zu ArrayList bspw.)
```

52 18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium

Open-Close Prinzip



- Offen für Erweiterung ⇔ Geschlossen für Veränderung
- Programmmodule sollen nicht verändert werden müssen

JavaDoc

00000

Exceptions

000000000000000

- Erweiterbaren Code schreiben
- Code soll nicht verändert werden

Übungsblatt 4

Péter Bohner: Tutorium 14

000000

Wiederholung

18.01.2023

000



Programmieren Tutorium

OO-Design-Prinzipien

00000000000

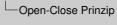
Java-API







Erweiterbaren Code schreiben



Code soll nicht verändert werden

Schnittstellenentwurf

18.01.2023



- Implementierungsdetails verbergen
- Konsistentes und angemessenen Abstraktionsniveau

Péter Bohner: Tutorium 14

Generelle Schnittstellen verwenden (void addUsers (Collection<User> users))

Übungsblatt 4 Wiederholung JavaDoc Java-API OO-Design-Prinzipien Exceptions 000 000000 00000 000000000000000 000000000 0000000000

Programmieren Tutorium







 Konsistentes und angemessenen Abstraktionsniveau Generalla Schnittstellen verwenden (vold additions (Collection-Users users))

-Schnittstellenentwurf

Weitere Prinzipien



- Niemals gleichzeitig in einer Methode Objekt-Zustand zurückgeben und ändern
- Idempotente Methoden schreiben (Hintereinanderausführung liefert konsistente Ergebnisse) ■ Richtige Methodennamen wählen (Beschreiben das Verhalten der Methode und sind konsistent)

Übungsblatt 4 Java-API OO-Design-Prinzipien Wiederholung JavaDoc Exceptions 000000 00000 000000000 0000000000

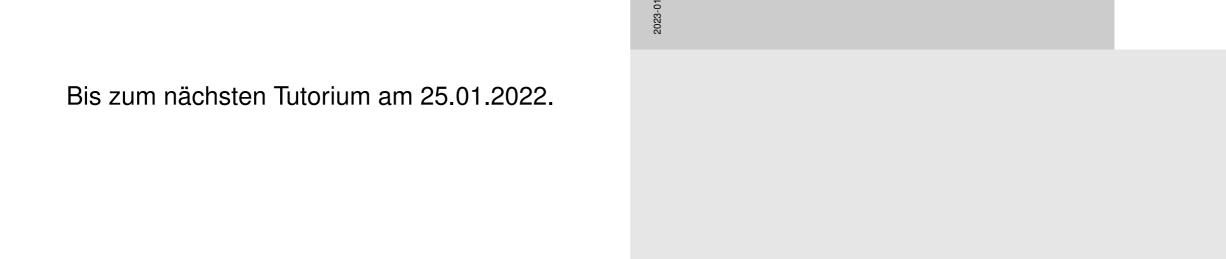
18.01.2023 Péter Bohner: Tutorium 14 Programmieren Tutorium Niemals gleichzeitig in einer Methode Obiekt-Zustand zurückgeben und ändern Idempotente Methoden achreiben (Hintereinanderausführung liefert konsistente Ergebnisse) Richtige Methodennamen w\u00e4hlen (Beschreiben das Verhalten der Methode und sind konsistent)

Weitere Prinzipien

OO-Design-Prinzipien

Tutorium 14

1. konsistente Methodennamen: add, remove, first, last usw. immer weiter verwenden



Tutorium 14

OO-Design-Prinzipien

Übungsblatt 4 OO-Design-Prinzipien Wiederholung JavaDoc Exceptions 000000 00000 000000000000000 000000000 0000000000 18.01.2023

Péter Bohner: Tutorium 14 Programmieren Tutorium

Bis zum nächsten Tutorium am 25.01.2022.