Karlsruher Institut für Technologie

7. Tutorium
Exceptions, Präsenzübung

Tutorium 14

Péter Bohner | 14.12.2022

Inhaltsverzeichnis

Wiederholung

Wie leitet man eine Klasse A ab?

```
class B extends A ...
```

Durch welches Schlüsselwort kann man die Ableitung einer Klasse verbieten?

```
final class C
```

Von wie vielen Klassen kann eine Klasse in Java erben?

1. Es gibt keine Mehrfachvererbung (multiple inheritance)

Was wird vererbt?

nicht-private Methoden, Attribute, und geschachtelte Klassen

Wie kann man eine Methode überschreiben?

`@Override` Mit identischer Methodensignatur.

Ohne @Override wird nicht sicher gestellt, dass tatsächlich überschrieben wird (stellt sicher, dass sonst ein Compilerfehler auftritt)

Wiederholung

Wie kann man prüfen, ob ein Objekt eine Instanz einer Klasse ist

```
object instanceof Class
```

Wie macht man einen Up-Cast?

```
Kraftfahrzeug k = new Motorrad(); // Harmlos!
```

Wie macht man einen Down-Cast?

```
Kraftfahrzeug k = new Motorrad();
if (k instanceof Motorrad) { // Immer mit Typenüberprüfung
    Motorrad m = (Motorrad) k;
}
```

Welches Schlüsselwort erlaubt eine unvollständige Implementation einer Klasse/Methode anzugeben?

```
abstract
```

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

Exceptions

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`**

Exceptions - Beispiel

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

Exceptions - Arten von Fehlern

Error

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

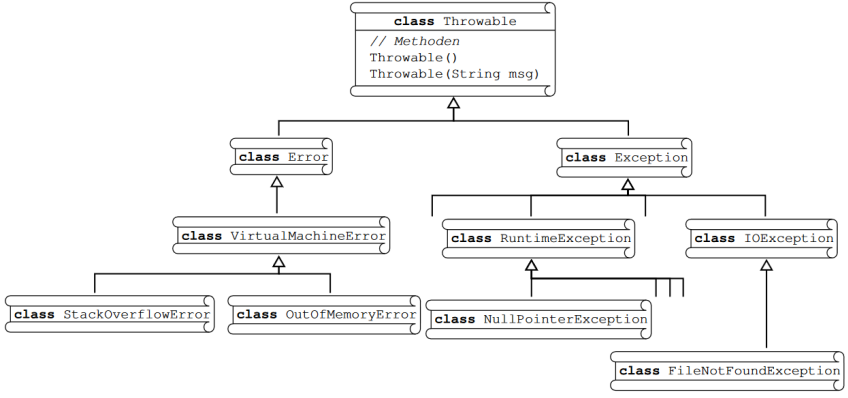
`RuntimeException`

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, ...

Exceptions - Hierarchie



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*

Exceptions

Beispiel

```java
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.");
}
```

Exceptions

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.

Exceptions

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 `IOException` Oberklasse von `FileNotFoundException` ist, müsste letzteres nicht extra deklariert werden. Dokumentationszwecke!

Exceptions

Ort der Behandlung

Finden der passenden Ausnahmebehandlung:

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

Exceptions - Konventionen

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

Exceptions - Konventionen

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();
}
```

Exceptions - Konventionen

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 - Konventionen

Verboten!

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

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 für dafür vorgesehene Zwecke (Javadoc anschauen)

Beispiele in der Java-API

- `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

Fehlererkennung

- so früh wie möglich
- defensiv
- mittels if Exceptions

Präsenzübung

Organisatorisches

- Am **13.01.2022 17:30-19:00**
- Nachholtermin erst im nächsten Semester
- Erfordelich für Übungsschein
- Keine Hilfsmittel
- KIT- **UND** amtlicher Lichtbildausweis
- Ihr seit im Tutorium **14**
- Hörsaaleinteilung im SDQ-NewsList Email-Verteiler `https://s.kit.edu/newslist`

Präsenzübung

Ihr müsst können

- Java-Grundlagen: Syntax, Überschattung, Klassen, Attribute, etc.
- Kontrollfluss: Schleifen, switch-case, if, etc.
- OOP-Grundlagen: Konstruktoren, Sichtbarkeit, Polymorphie

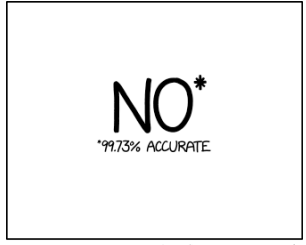
Tipps

- Die größe der Lücken hat keine Aussagekraft
- eure Lösungen müssen nicht schön sein, nur funktionieren.
- Die (letzte) Polymorphie/dynamische Bindungsaufgabe ist die schwerste.
- Im ILIAS Ordner gibt es diese Hinweise und eine Probe-Präsenzübung, die ihr noch selber machen könnt

Bis zum nächsten Tutorium am 11.01.2022.

Ab dem Mittwoch gibt es keine Tutorien mehr!

Is it Christmas yet?



XKCD.COM PRESENTS A NEW "IS IT CHRISTMAS" SERVICE TO COMPETE WITH ISITCHRISTMAS.COM

We've tested it on 30 different days and it hasn't gotten one wrong yet.