ingenieur
wissenschaften
htw saar

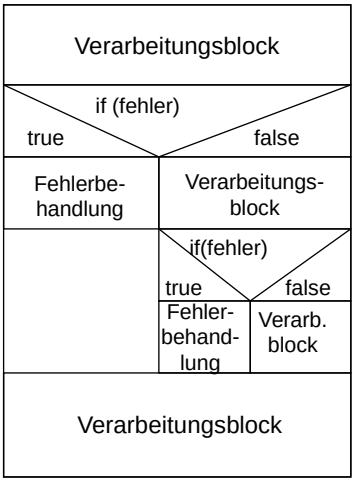
Hochschule für
Technik und Wirtschaft
des Saarlandes
University of
Applied Sciences

Einführung Ausnahmebehandlung

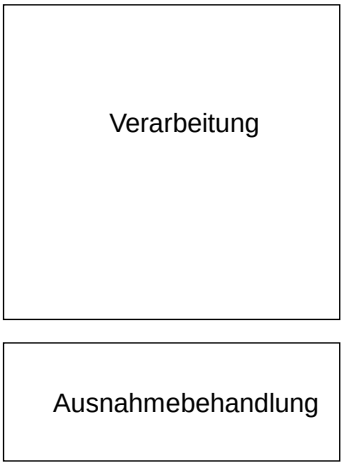
Prof. Dr. H. G. Folz

Gegenüberstellung

Klassische Fehlerbehandlung



Ausnahmebehandlung



Sprachelemente der Ausnahmebehandlung

`throw` *`Ausnahmeobjekt`*

- ⇨ Auswerfen einer Ausnahme.
- ⇨ Die Programmsteuerung wird an die nächste zuständige Ausnahmebehandlung übergeben, sofern eine vorhanden ist.
- ⇨ Ist keine Ausnahmebehandlung vorhanden, so wird das Programm abgebrochen.

Sprachelemente der Ausnahmebehandlung

```
try {
    zusammengesetzte Anweisung
} Folge von Ausnahmebehandlungsroutinen
```

- ⇨ Ein `try-Block` fasst Anweisungen zusammen, für die eine Ausnahmebehandlung durchgeführt werden soll.

```
catch (Ausnahmedeklaration) {
    zusammengesetzte Anweisung
}
```

- ⇨ Ausnahmebehandlungsroutine
- ⇨ Fängt Ausnahmen eines bestimmten Typs auf.

Sprachelemente der Ausnahmebehandlung

```
finally { zusammengesetzte Anweisung }
```

- Optionale abschließende Anweisungen nach dem `try-Block`
- Der Code im `finally-Block` wird garantiert ausgeführt, sobald nur eine Anweisung des `try-Blocks` ausgeführt wurde

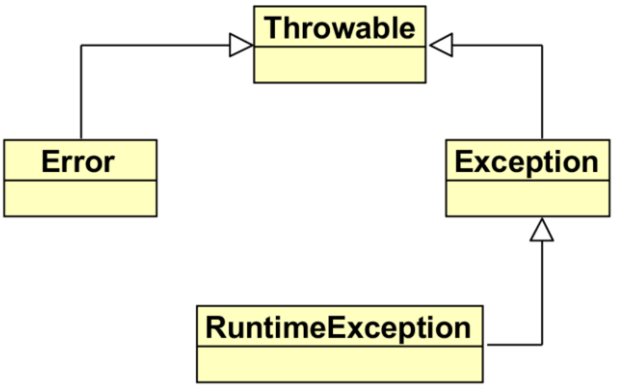
Sprachelemente der Ausnahmebehandlung

```
try {
    Anweisungen und Ausdrücke, die Ausnahmen auslösen
   können  z. B.
           throw new Ausnahmetyp1();

}
catch (Ausnahmetyp1 a) {
    fängt Ausnahmen der Klasse Ausnahmetyp1 auf und
    kann sie behandeln
}
catch (Ausnahmetyp2 a) {
    fängt Ausnahmen der Klasse Ausnahmetyp2 auf und
    kann sie behandeln
}
finally {
    Abschlussaktionen nach dem Try-Block...
}
```

Die Java-Ausnahmeklassen

Alle Ausnahmen in Java sind Unterklassen der Klasse `java.lang.Throwable`.
`Throwable` ist eine allgemeine Ausnahmeklasse, die im wesentlichen eine Klartext-Fehlermeldung speichern und einen Auszug des Laufzeit-Stacks ausgeben kann.



Wichtiges zur Anwendung

- `catch (RuntimeException e)`
  fängt alle Laufzeit-Ausnahmen auf.
- `catch (Exception e)`
  fängt alle Ausnahmen auf, in deren Name das Wort `Exception` vorkommt.
- `catch (Error e)`
  fängt alle Ausnahmen auf, in deren Name das Wort `Error` vorkommt.
- `catch (Throwable e)`
  fängt alle Ausnahmen auf.
- `printStackTrace()`
  gibt den sogenannten Ausführungsstack auf die Standardfehlerausgabe System.err aus.
- `printStackTrace(System.out)`
  gibt den sogenannten Ausführungsstack auf die Standardausgabe `System.out` aus.

Was ist eigentlich assert?

`assert (Bedingung) : Objekt`
entspricht etwa der folgenden Anweisung

```
if (!Bedingung) {
    throw new AssertionError(Objekt);
}
```

- Was ist der Unterschied?
- Wann wendet man **`assert`** an und wann direkt **`throw`**?