UNIVERSITÄT
LEIPZIG

Einführung in die Objekt-Orientierte Modellierung und Programmierung

Wintersemester 2025/2026

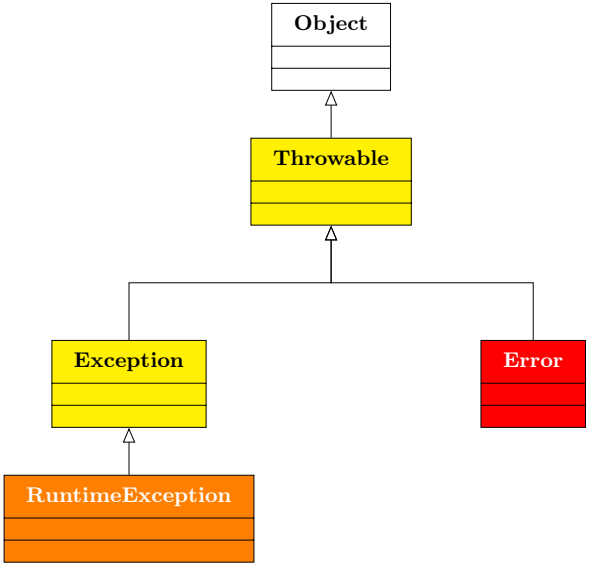
Dirk Zeckzer

Institut für Informatik

Teil XVI

Ausnahme- und Fehlerbehandlung

Throwable



Gelb	Checked Exception
Orange	Unchecked Exception
Rot	Error

Throwable

Nur die Konstruktoren sind überladen.

- `Throwable ( )`
- `Throwable ( String message )`

Methoden

- `String getMessage()`
- `void printStackTrace()`
- `String toString()`

Throwable fangen

```
1  try {
2      <statements>
3  } catch ( Throwable thr ) {
4      <tue etwas sinnvolles>
5  } finally {
6      <aufraeumen>
7  }
```

Throwable fangen

Mehrere Throwable fangen

- von speziell zu allgemein
- Beispiel:
  1. NullPointerException
  2. RuntimeException
  3. Exception

```
1   try {
2       <statements>
3   } catch ( NullPointerException npEx ) {
4       // tue etwas sinnvolles fuer NullPointerException
5   } catch ( RuntimeException rtEx ) {
6       // tue etwas sinnvolles fuer sonstige RuntimeException
7   } catch ( Exception ex ) {
8       // tue etwas sinnvolles fuer sonstige Exception
9   } finally {
10      <aufraeumen>
11  }
```

Eigenes Throwable erstellen

```
1  public class FalscherWertFuerRadiusException
2    extends Exception {
3
4    private final double radius;
5
6    public FalscherWertFuerRadiusException(
7      final double radius
8    ) {
9      super("Falscher Wert fuer Radius: r = " + radius);
10     this.radius = radius;
11   }
12
13   public double getRadius() {
14     return radius;
15   }
16 }
```

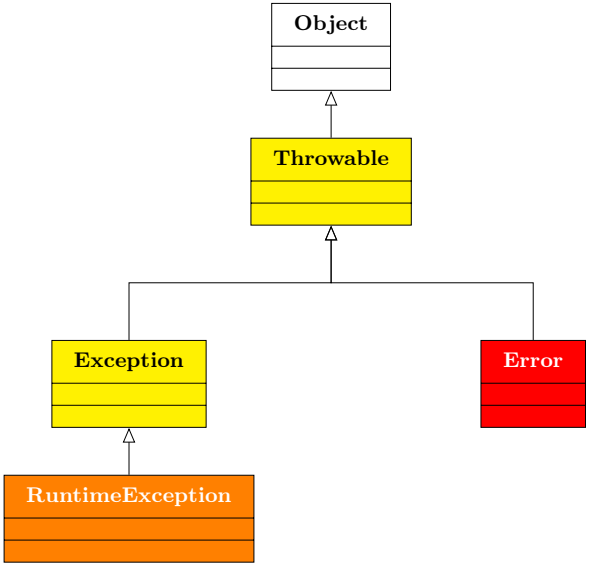
Eigenes Throwable werfen

```
1   public class KreisMitException {
2
3     private static final double PI = 3.1415926536;
4
5     private double radius;
6
7     public KreisMitException() {
8       radius = 1.0;
9     }
10
11    public KreisMitException(
12      double radius
13    ) throws FalscherWertFuerRadiusException {
14      if (radius > 0) {
15        this.radius = radius;
16      } else {
17        throw new FalscherWertFuerRadiusException(radius);
18      }
19    }
20
21    ...
22  }
```

Eigenes Throwable fangen

```
1   public class KreisMitExceptionMain {
2
3     public static void main(
4       final String[] args
5     ) {
6       KreisMitException kreis = new KreisMitException();
7       try {
8         kreis = new KreisMitException(-21.1);
9       } catch (FalscherWertFuerRadiusException falscherWertFuerRadiusException) {
10        System.err.println(falscherWertFuerRadiusException.getMessage());
11        try {
12          kreis = new KreisMitException(-falscherWertFuerRadiusException.getRadius());
13        } catch (FalscherWertFuerRadiusException falscherWertFuerRadiusException2) {
14          System.err.println(falscherWertFuerRadiusException2.getMessage());
15        }
16      }
17
18      System.out.println("Instanz: Kreis mit Radius r = " + kreis.getRadius());
19    }
20  }
```

Throwable



Gelb	Checked Exception
Orange	Unchecked Exception
Rot	Error

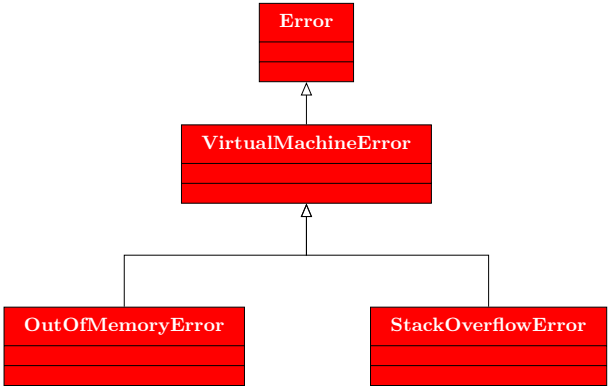
Throwable

Type	Property	Sollte gefangen werden	Catch oder Throws notwendig

Checked Exception	`Throwable`
	`Exception`
	Erweiterungen von `Exception`
	aber nicht von `RuntimeException`
		Ja	Ja

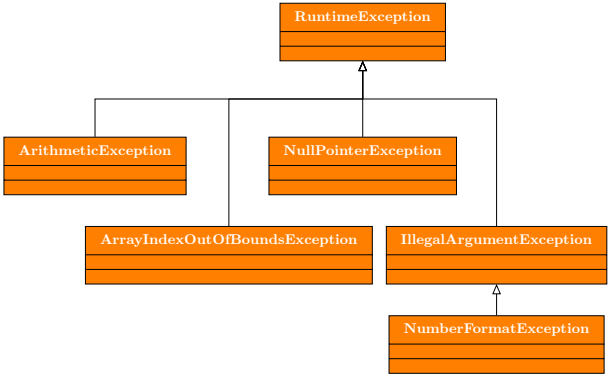
Unchecked Exception	`RuntimeException` Erweiterungen von `RuntimeException`	Ja	Nein
Error	`Error` Erweiterungen von `Error`	Nein	Nein

Error



Gelb	Checked Exception
Orange	Unchecked Exception
Rot	Error

RuntimeException (Checked Exception)



Gelb	Checked Exception
Orange	Unchecked Exception
Rot	Error