Die try-catch-Anweisung

```
try {
   Anweisung;
   ...
} catch (Ausnahmetyp1 x) {
     Anweisung;
   ...
}
...
} catch (Ausnahmetypn x) {
     Anweisung;
   ...
}
```

Hier kann Ausnahme auftreten

Behandlung der Ausnahme vom Typ 1

Behandlung der Ausnahme vom Typ n

Bsp.: Ausnahmebehandlung

```
public class AusnahmeBsp {

  public static void main(String[] args) {
    int i;

// Basis der Darstellung von 40:
    int base = 0;

// Dezimalwert von 40 zu Basen 10,..,2:
    for (base = 10; base >= 2; --base) {
      i = Integer.parseInt("40",base);
      System.out.println("40 base "+base+" = "+i);
    }
  }
}
```

JDK-Ausgabe ohne Ausnahmebeh.

```
java.lang.NumberFormatException: 40
        at java.lang.Integer.parseInt(Integer.java:414)
        at übungzuv5.AusnahmeBsp.main(AusnahmeBsp.java:22)
40 base 10 = 40
40 base 9 = 36
40 base 8 = 32
40 base 7 = 28
40 base 6 = 24
40 base 5 = 20
Exception in thread "main"
```

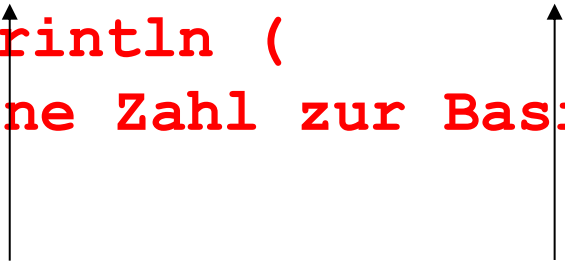
Problem? 40 ist keine gültige Zahl zur Basis 4 und kleiner

Abfangen der Ausnahme

```
try {

  for (base = 10; base >= 2; --base) {
    i = Integer.parseInt("40",base);
    System.out.println("40 base "+base+" = "+i);
  }

} catch (NumberFormatException e) {
    System.out.println (
      "40 ist keine Zahl zur Basis "+ base);
}
```



JDK-Ausgabe mit Ausnahmebehandlung

40 base 10 = 40

40 base 9 = 36

40 base 8 = 32

40 base 7 = 28

40 base 6 = 24

40 base 5 = 20

40 ist keine Zahl zur Basis 4

nicht mehr Exceptionmeldung,
sondern Ausgabe der Ausnahmebehandlung

Fortfahren nach Ausnahmebehandlung

```
for (base = 10; base >= 2; --base) {
  try {
    i = Integer.parseInt("40",base);
    System.out.println("40 base "+base+" = "+i);

  } catch (NumberFormatException e) {
      System.out.println (
       "40 ist keine Zahl zur Basis "+ base);
  }
}
```

Ausgabe des JDK bei Fortfahren

40 base 10 = 40

40 base 9 = 36

40 base 8 = 32

40 base 7 = 28

40 base 6 = 24

40 base 5 = 20

40 ist keine Zahl zur Basis 4

40 ist keine Zahl zur Basis 3

40 ist keine Zahl zur Basis 2

verschiedene Ausnahmen

```
String[] numbers = new String[3];
numbers[0] = "10";
numbers[1] = "20";
numbers[2] = "30";
try {
  for (int base = 10; base >= 2; --base) {
    for (int j = 0; j <= 3; ++j) {
      int i = Integer.parseInt(numbers[j],base);
      System.out.println(numbers[j]+"base "+base+" = "+i);
    }
  }
} catch (IndexOutOfBoundsException e1) {
    System.out.println("***IOOBEx: " );
} catch (NumberFormatException e2) {
    System.out.println("***NFEx: " );
}
```

Feldüberlauf

mehrere Ausnahmen (JDK-Ausgabe)

10 base 10 = 10

20 base 10 = 20

30 base 10 = 30

***IOOBEx:

Verwendung der `finally`-Klausel

```
try {
   for (base = 10; base >= 2; --base) {
     i = Integer.parseInt("40",base);
     System.out.println("40 base "+base+" = "+i);
   }

} catch (NumberFormatException e) {
     System.out.println (
       "40 ist keine Zahl zur Basis "+ base);

} finally {
     System.out.println("Schluss!");
}
```

JDK-Ausgabe mit `finally`-Klausel

```
40 base 10 = 40
40 base 9 = 36
40 base 8 = 32
40 base 7 = 28
40 base 6 = 24
40 base 5 = 20
40 ist keine Zahl zur Basis 4
Schluss!
```

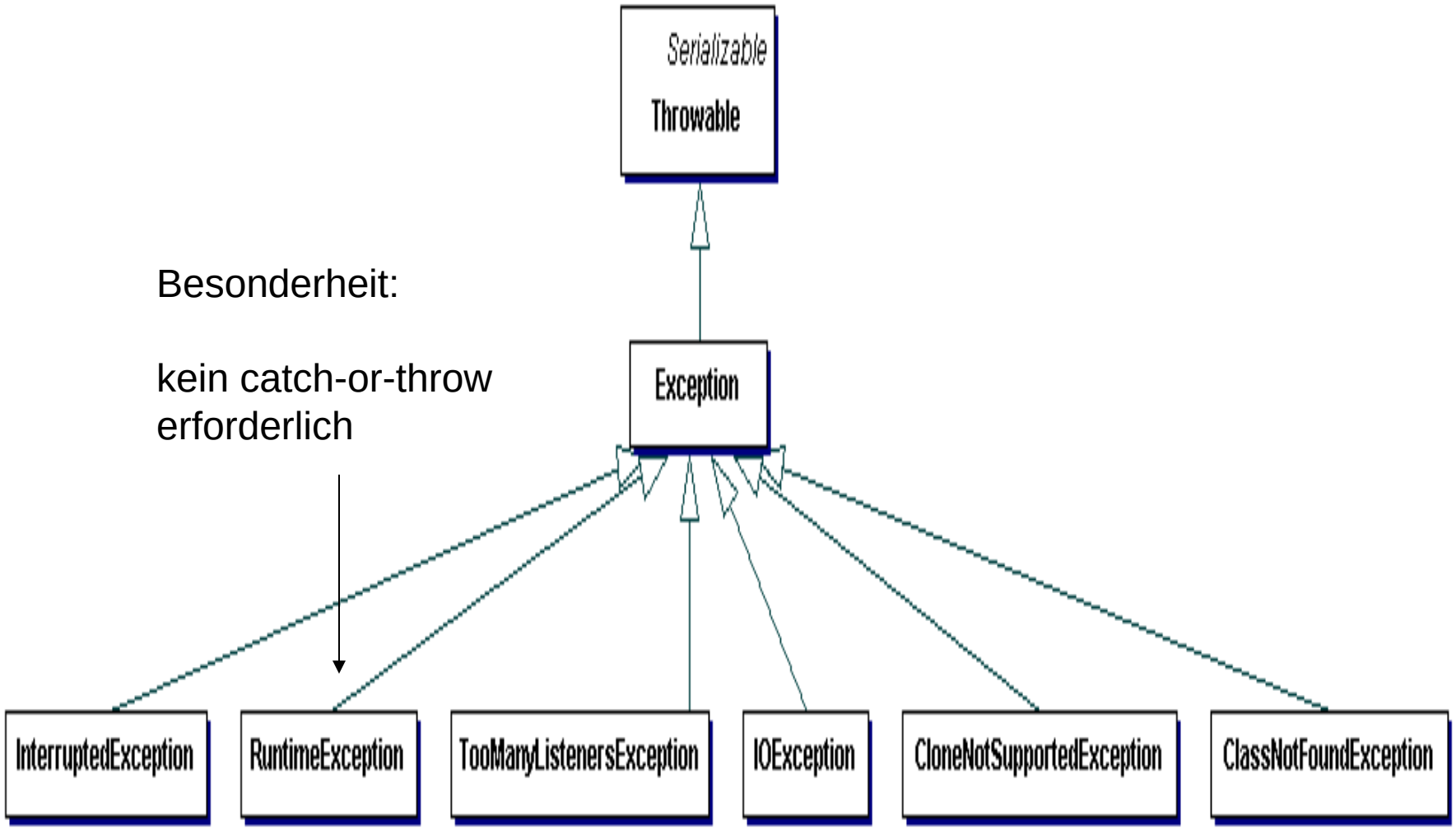
Weitergabe einer Ausnahme (`throws`)

```
public void reziTable()throws ArithmeticException
{
  int x = 0;
  while (x <= 10) {
    System.out.println("rezi("+x+")="+(double)(1/x));
    x++;
  }
}
```


hier kann Ausnahme auftreten

nicht behandelt,
darum weitergeben mit throws

Die Exception-Hierarchie



Auslösen einer Ausnahme mit `throw`

```
public class Auslösen {
  public void nurPositiv(int zahl)
              throws ArithmeticException {
    if (zahl <= 0) {
      throw new ArithmeticException(
            "Parameter nicht positiv!");
    }
  }
  public static void main(String[] args) {
    Auslösen auslösen = new Auslösen();
    auslösen.nurPositiv(0);
  }
}
```

```
java.lang.ArithmeticException: Parameter nicht positiv!
        at übungzuv5.Auslösen.nurPositiv(Auslösen.java:6)
        at übungzuv5.Auslösen.main(Auslösen.java:11)
Exception in thread "main"
```

Eigene Exception-Klasse verwenden

```
public class EE {
  public void nurPositiv(int zahl)
              throws NegativerParameterException {
    if (zahl <= 0) {
      throw new NegativerParameterException(
            "Parameter nicht positiv!");
    }
  }
  public static void main(String[] args) {
    EE ee = new EE();
    ee.nurPositiv(0);
  }
}
```

```
Exception in thread "main" NegativerParameterException:
Parameter nicht positiv!
at EE.nurPositiv(EE.java:11)
at EE.main(EE.java:16)
```

Eigene Exception-Klasse definieren

```
public class NegativerParameterException extends
ArithmeticException {
        NegativerParameterException(String ausgabe) {
                super(ausgabe);
        }
}
```


Konstruktor implementieren
durch Aufruf des Oberklassenkonstruktors

```
public static void main(String[] args) {
EE ee = new EE();
try {
    ee.nurPositiv(0);
  } catch (NegativerParameterException e) {
        e.printStackTrace();
      }
}
```