Der `noexcept` Spezifizierer

Man darf (und soll) bei Funktionen und Methoden, die ganz sicher keine Exceptions werfen (und auch nichts aufrufen, was selbst Exceptions werfen könnte!) das **`noexcept`** Keyword nachstellen. C++ darf in diesem Fall aggressiver optimieren und das Programm kann schneller werden. Kommt es dennoch zu einer Exception aus diesem Codeteil, so ist das nicht mehr durch einen `try-catch` Block abzufangen sondern führt immer zum Programmabbruch! Dieses Keywort ist Teil der Funktionssignatur, d.h. Funktionen, die sich nur in diesem Keywort unterscheiden, sind verschieden! Insbesondere muss bei virtuellen Methoden dieses Keywort in allen Klassen identisch vorhanden sein (überall oder nirgends).

```
… f(…)                {…}                    // darf Exceptions werfen
… f(…) noexcept       {…}                    // darf keine Exceptions werfen
… f(…) const                    {…}          // darf Exceptions werfen
… f(…) const noexcept           {…}          // darf keine Exceptions werfen
```

Manchmal findet man (vor allem in hochoptimierten Bibliotheken) komplexere Anwendungen von `noexcept`. Man kann nämlich zusätzlich Bedingungen formulieren, wann eine Funktion garantiert keine Exception wirft (conditional noexcept).

Folgende Codeteile **sollten niemals Exceptions werfen** und sollten daher immer als `noexcept` gekennzeichnet werden (die von C++ erstellten speziellen Methoden sind automatisch so spezifiziert, wenn das möglich ist!):

1) Destruktoren
2) move-Konstruktoren
3) move-Zuweisungsoperatoren